R-977

INERTIAL NAVIGATION SYSTEM STANDARDIZED SOFTWARE DEVELOPMENT

FINAL TECHNICAL REPORT

Volume III of IV

PROGRAM DESCRIPTION AND USER'S GUIDE

June 1976

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER R-977	2. GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INERTIAL NAVIGATION SYSTEM STANDARDIZED SOFTWARE DEVELOPMENT, FINAL TECHNICAL REPORT, VOLUME III, PROGRAM DESCRIPTION & USER'S GUIDE		5. TYPE OF REPORT & PERIOD COVERED 3/15/76-4/30/76 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. Sciegienny, R. Nurse, J. Wexler, P. Kampion		8 CONTRACT OR GRANT NUMBER(s) F33615-75-C-1149
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Charles Stark Draper Laboratory, Inc. 68 Albany Street Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS Task 4.2.1
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory Wright-Patterson Air Force Base Dayton, Ohio 45433		12. REPORT DATE June 1976 13 NUMBER OF PAGES
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15 SECURITY CLASS (of this report) UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

NTIS	White Section X
DDC	Buff Section ☐
UNANNOUNCED	☐
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL and/or SPECIAL
A	

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Inertial Navigation
Computer Algorithms
Computation Errors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Section 1 of Volume III provides an overview of the purpose, structure, capabilities, and operation of the Numerical Simulator Program (NUMSIM) as of 30 April 1976.

Section 2 contains the detailed description of NUMSIM. First, the symbology, conventions, and variables are defined. Then the individual routines are described under the headings of: function and equations; input, output, and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

internal variables; and finally the flow chart of the routine. This sectio- is the only place where the description of the manipulations of the profile generator (PROFGEN) inputs to the form required by the particular "ideal" inertial measuring unit (IMU) outputs is provided.

Section 3 describes the operation of NUMSIM in greater detail, under the headings of: operator inputs; simulator outputs; and operational aids. The use of PROFGEN in conjunction with NUMSIM is also described.

The appendices contain sample input and output data, file structure and control cards.

It is intended that this volume be studied together with program listings in Volume IV.

R-977

INERTIAL NAVIGATION SYSTEM STANDARDIZED SOFTWARE DEVELOPMENT

FINAL TECHNICAL REPORT

VOLUME III

Program Description

and

User's Guide

June 1976

Approved: W. G. Denhard

The Charles Stark Draper Laboratory, Inc.

Cambridge, Massachusetts 02139

ACKNOWLEDGEMENT

This four-volume report was prepared under USAF Contract F33615-75-C-1149 by Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts, in accordance with Section 4 of the contract. The monitoring Air Force project engineer is Captain E. Harrington (RWA-2), Air Force Avionics Laboratory, Dayton, Ohio.

The Draper Laboratory Program Manager for this task is Dr George T. Schmidt and the Lead Engineer is Arthur Ciccolo. The coordinator of this report is Janusz Sciegienny and the authors are Janusz Sciegienny, Roy Nurse, Peter Kampion and John Wexler.

The authors express their appreciation to Mr. W. Shephard, Mr. D. Kaiser and Mr. Stan Musick of the Air Force Avionics Laboratory for their assistance during the course of this contract.

VOLUME III

Simulator Program Design

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

List of Illustrations

1.0 INTRODUCTION

1.1 Purpose of Function

a) Statement of Problem

This volume contains a detailed description of the simulator program design, developed in this task study.

The purpose of this simulator program is to allow the determination of the Inertial Navigation System (INS) navigation errors in position, velocity and attitude resulting from mechanization of the navigation equations on a digital computer for a "perfect" Inertial Measurement Unit (IMU). A perfect IMU consists of error free instruments, that is the uncompensated gyros are not drifting and sense the true inertial angular velocity, the uncompensated accelerometers sense the true specific force and the gimbal angle resolvers generate the true angles.

The program provides simulation of the navigation computation, for three different IMU mechanizations, namely a local level wander azimuth platform stabilization, a space stabilized platform and a strapdown configuration. All three mechanizations use a perfect barometric altimeter. No other navigation aids are employed in the computations.

The simulated navigation computations are performed in a local-level, wander azimuth computational frame.

The computations are assumed to be performed on an airborne digital computer using a "floating point" representation of a specific length common to all variables.

b) Simulated Errors

The effect of the following error sources can be investigated.

i) Quantizations of

- A) Accelerometer inputs
- B) Gimbal angle resolver or incremental angle inputs
- C) Torquing commands to local level IMU

ii) Vertical channel damping mechanization - second or third order, value of parameters

iii) Iteration rates, shorter versus longer computational cycles

iv) Word Length, that is the precision to which the airborne computer is performing its calculations

v) Navigation Algorithm implementation:
   A) "Order" of navigation equations
   B) Time between orthormalizations of direction cosine matrix ('DCM')
   C) Formulas used in calculations of angular velocity
   D) Order of Direction Cosine Matrix (DCM) updates

vi) Library routine implementation - manner in which sine/cosine, arctangent and square root routines are implemented

vii) IMU type used in the simulation
   A) Local level, wander azimuth platform
   B) Space stable platform
   C) Strapdown - in this configuration one may also investigate
      I) Effect of quaternion vs. DCM implementation
      II) Effect of different update orders, or orthonormalization rates.

The effect of different gravity models on the navigation errors is not included in this simulation.

One or more of these error sources can be introduced simultaneously, in order to investigate the effects of combinations of errors.

c) Description of Inputs, Outputs and Usage

To use the simulator to investigate the effects of given computation errors on a given mission's navigation errors one must:

i) Specify the "flight profile" for AFAL's supplied flight profile generator computer program PROFGEN.

ii) Run the profile through PROFGEN, producing an output file (a magnetic tape presumably) containing the specific forces and gimbal angles that would be

sensed during the mission, along with the position the velocity and the attitude.

iii) Establish the baseline for comparison either by:
   A) using the PROFGEN output, or by
   B) running the simulator with no intentional computation errors, that is, full precision, no quantization, etc.

iv) Specify the configuration to the simulator and run the simulator using the file from step ii) as input. This will create an output file (a magnetic tape presumably) with the computed position, velocity, and attitude.

v) If one used iii B), difference the output of iv) against the baseline.

vi) Perform post processing of the differences (computation of the RMS navigation errors induced by various computation errors,etc.)

The inputs, outputs and method of operation are described in more detail in Section III of this volume.

In brief, the inputs consist of specifying numerical values for the errors (e.g. the number of bits of precision) and numerical values for various control parameters (e.g. how often to print results). The outputs are both an output file and printed output. The output file contains the computed position, the velocity and the altitude, and the differences between these computed values, and the position, the velocity and the attitude generated by PROFGEN. The printed output also has these values and can have, in addition, various "debugging" diagnostic printouts.

1.2 Overview: Functional and Structural

a) Simulator Design

A functional block diagram of the simulator program is shown in Figure 1. The simulator consists of the following parts:

1. A flight profile generator (PROFGEN)
2. An IMU simulator
3. A navigation computer simulator
4. An error processor.

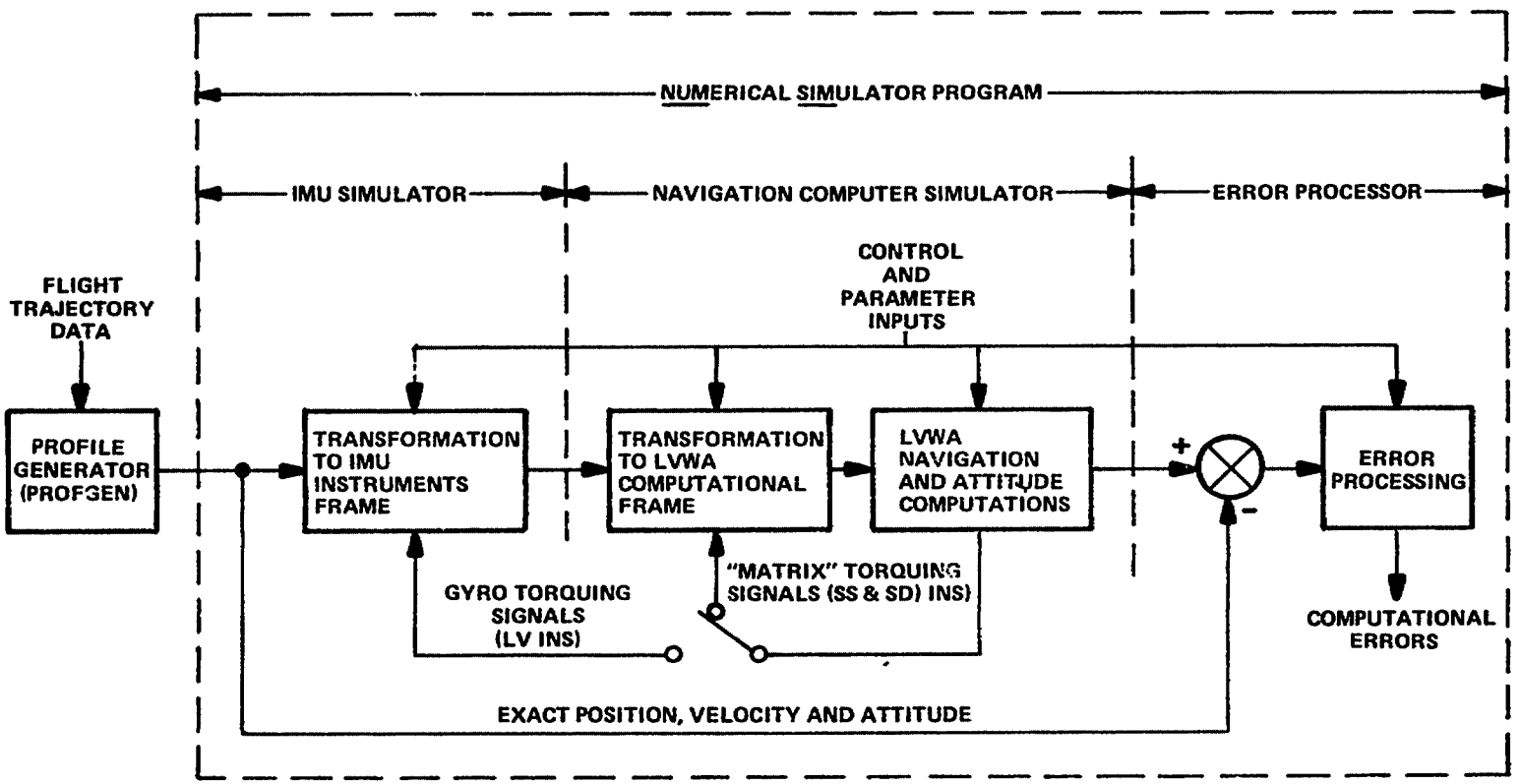


Figure 1. Numerical Simulator Functional Block Diagram

The PROFGEN generates the specific forces, the aircraft attitude, the position and the velocity for a given flight trajectory. The specific forces and the attitude data are transformed by the IMU simulator from the PROFGEN computational frame (local level, geodetic, wander azimuth) to the IMU instruments frame. The data is then transformed in the navigation computer simulator to the navigation computer computational frame (local level, geodetic, wander azimuth) and the aircraft position, velocity and attitude are computed on the simulated aircraft navigation computer. The computed navigation data is compared with the PROFGEN generated navigation data in the error processor. The error processor outputs are the instanteous navigation errors in the computed position, velocity and attitude caused by mechanization of the navigation equations on an airborne digital computer for a specified flight mission.

b) Calculations Performed

Several preliminary concepts must be set forth before examining the calculations performed. The first is that of a 'cycle' as it appears in the terms 'attitude cycle', 'navigation cycle', 'filter cycle' and 'sample cycle'. The sample cycle refers to the time between inputs from the PROFGEN output file; as an example if there are 128 outputs per second, then the sample cycle is 1/128 sec. The attitude cycle is the time between attitude calculations in the simulated airborne computer; as an example, if the airborne computer computes attitude 64 times per second, then the attitude cycle is 1/64 sec. Similarly, there are cycle times defined for the attitude filter (which performs a statistical filtering of the attitude data) and for the navigation calculations (which computes position and velocity).

The simulator is set up so that these cycles must bear certain relationships to one another. To be specific, the attitude cycle must be an integral multiple of sample cycles, the filter cycle must be an integral number of attitude cycles, and the navigation cycle must be an integral number of filter cycles.

The second concept is that of "IMU simulator" coding versus "flight code" coding. The "flight code" consists of that software which would actually be programmed in to the airborne computer.

In brief,this coding takes the accelerometer data (the delta= velocities) and attitude information, processes it, and produces the navigation data (the velocity, the position and the attitude).

The "IMU simulator", on the other hand, accepts information from PROFGEN and performs those calculations necessary to generate the inputs to the flight code. The IMU simulator takes into account

i) The PROFGEN sample cycle being different from either the attitude, filter, or navigation cycles.

ii) Misalignments between the actual platform and the flight codes knowledge of platform alignment.

iii) An IMU orientation other than the local level; local level is used by PROFGEN.

The third, and final, preliminary concept is that of IMU stabilization, which describes the orientation that the accelerometers and attitude sensors will have during the mission. Three options are available:

i) Local Level Wander Azimuth - in this configuration the platform is 'torqued' (rotated with respect to inertial space) in such a way as to keep two of the axes level (level is defined as being perpendicular to the geodetic gravity vector), and to compensate for the earth rate. The effect of this is that if the IMU were stationary on some point on the earth, an observer at that point would not see a rotation about any of the axes. Any easterly motion of the IMU at non-zero latitude, would cause a rotation in azimuth (that is, about the up axis) to be seen by an observer riding along with the IMU.

ii) Space Stable - in this configuration the platform maintains a constant orientation with respect to inertial space.

iii) Strapdown - in this configuration the accelerometers are rigidly attached to the body of the vehicle so that the accelerometers rotate (with respect to inertial space) the same way that the body does.

For the local level case, the IMU simulator forms up the attitude angles once every DELTA. The specific forces are accumulated over DELT (the navigation cycle). During the attitude cycle in which the navigation calculations are to be performed, the navigation calculations are performed prior to the attitude calculations. The attitude filter is run every DELTF (the filter cycle) after the attitude computations for that attitude cycle.

The space stable case is nearly identical, the sole difference being that, just prior to the navigation calculations, calculations are performed to rotate the delta-velocity (which is coming from a simulated space stable IMU) into the local level wander azimuth computational frame.

The calculations performed, for the local-level wander azimuth IMU, and the space stable IMU are shown in Figure 2.

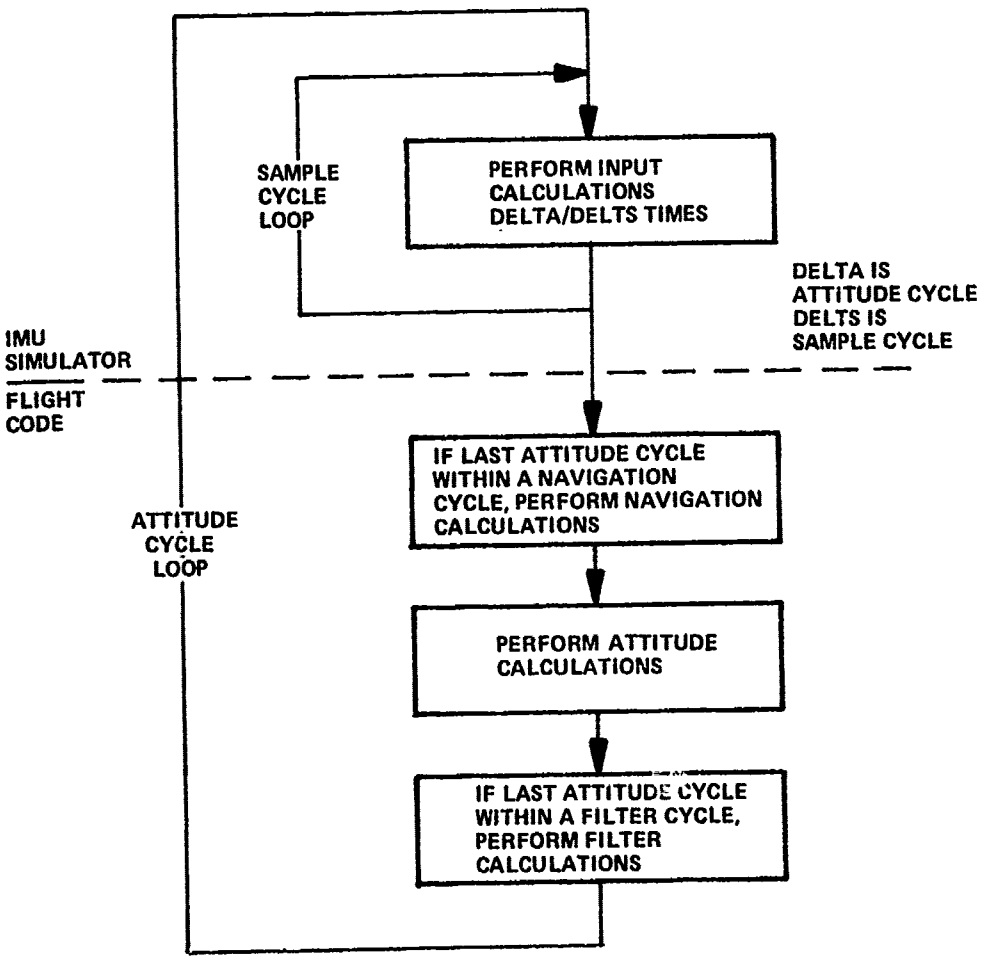


Figure 2. Local Level/Space Stable Calculations

The calculations performed for the strapdown IMU are similar, and are shown in Figure 3.

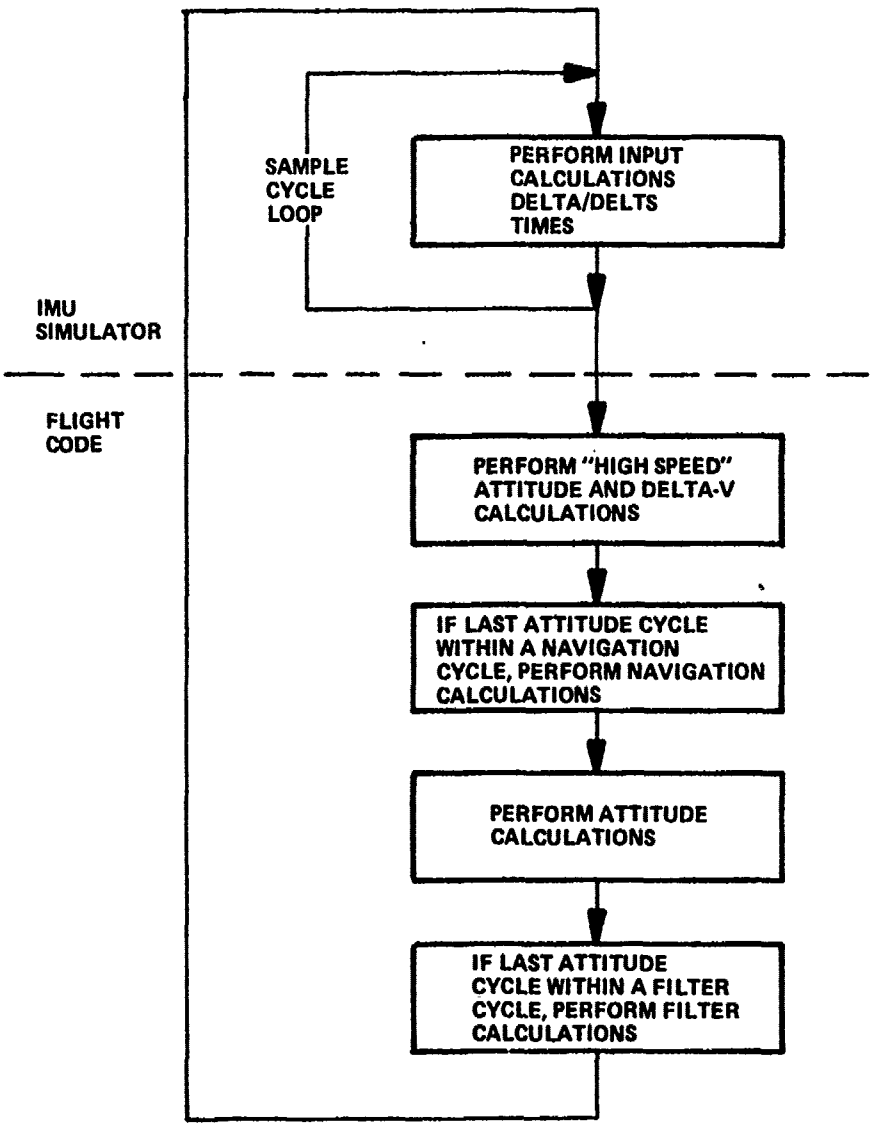


Figure 3. Strapdown Calculations

The strapdown calculations are also similar to the local level. The difference is the existence of the "high-speed" delta-v summation, which is run every attitude cycle. The purpose of this summation is to accumulate the delta-v in the computation frame. This high speed summation is necessitated by the high angular rates (of body with respect to computational frame) achieved.

c) "Flight Code"

Flight code, as discussed before, consists of those calculations that would be performed in an airborne computer. For the purposes of this simulator, it is assumed that all quantities are stored in a floating point format. It is further assumed that the amount of storage for each quantity is the same, that is, each quantity is computed to the same precision.

Since the flight code computations will be performed in real-time (that is, synchronized with events in the real (physical) world), the goal of the design of the computational algorithms is to achieve the maximum computational precision (minimize the maximum error due to the approximations selected) while keeping the computational load (that is, maximum computation time per cycle) below some fixed constraint.

d) Initialization

The initialization performed can be viewed in two parts. First is the setting of control variables regulating cycles, printout selection, etc. Second is the alignment of the perfect IMU and initialization of the navigation algorithms that would take place in the airborne computer during alignment.

This initialization is performed by the 'executive' (the main program), the BLOCK DATA subprogram, and the various intialization subroutines.

e) Modularity and Interroutine Communication

The simulator is constructed of a main program and approximately 51 subroutines. The subroutines can be divided into four categories:

1) Initialization subroutines

2) IMU simulator subroutines (including all input and output routines).

3) Flight code subroutines.

4) Library subroutines (includes trigonometric functions, matrix multiply, etc.).

The main program (also called the 'executive' or the 'exec') calls, in the proper sequence, the initialization subroutines, the IMU simulator subroutines and the flight code subroutines. The library subroutines are made use of by the other three categories of subroutines. The interrelationship of subroutine is displayed graphically in the appendix "Subroutine Call Structure".

The individual subroutines were designed to be as modular (that is, performing a well defined role independently of the actions of other subroutines) as possible. The interroutine communication is handled as follows:

1) All variables are kept in labelled COMMON blocks.

2) Certain of the lower-level subroutines are 'passed' variables through an argument list.

1.3 Operation

a) Method of Operation

The simulator is setup to run on the CYBER 74, under the control of either SCOPE or INTERCOM. The user of the simulator (called the "operator" throughout) must have the PROFGEN binary output (also called "output tape") available to the program. In addition, the operator must prepare input cards as described in the "Program Operation" section of this documentation. The "Local Files" necessary to run the program are described in an appendix.

b) Inputs & Outputs

The inputs were described in brief in Section 1.2. The outputs conist of

i) the magnitude of the errors at the end of the simulated mission and

ii) a binary output tape (suitable for post processing) with the differences ('errors') in position, attitude and velocity.

c) Interpretation of Results

One needs to examine both 1) the magnitude of the errors at the end of the simulated mission and ii) the RMS (with respect to time) errors in order to be able to evaluate the performance of a given (simulated) system during the mission. All such post processing must be programmed by the operator.

2.0 DETAILED PROGRAM DESCRIPTION

2.1 Introduction

This section contains all the detail information related to the main program and all the subroutines; for each routine there is:

a) A discussion of the function and equations of the routine.
b) A list of the inputs, outputs and internal variables.
c) A functional flowchart, where applicable.

Note that

a) Calls to the DOUT (debugging output) routine are not shown or discussed.
b) In the flight code section
   i) Calls to the truncation routine, VP, are not shown
   ii) Calls to VSQRT, VMM etc. are shown as calls to SQRT, MM, etc.
c) The arrays A & B, when listed as internal variables, actually use the A & B of the NAVCOM COMMON block as a work area.

Figure 4 of this section contains the symbol list for all variables in COMMON, excluding only those variables (whose nature is fairly trivial) found in the OUTPUT, UNVRSL and PGRAV common block.

With regard to the symbol listing, note the following:

1) Except as noted, all variables are used for all three (local level 'LL', space stable 'SS', and strapdown 'SD' IMU types.
2) Frames (except as listed) used in this program are defined in Volume II, Appendix A of this report.
   a) 21 is the frame of the accelerometer axes
   b) 5Q is PROFGENs body frame
   c) 2QP is PROFGENs computational frame
3) Symbol XXINTG designates LLINTG, SSINTG or SDINTG, which ever is appropriate. Similar designation is used for symbols XXATUD, XXINIT. Symbol XXATT referes to LLATT, SSATT and a portion of HSINTG.

4) For the purposes of this section, ANGVL2 and TORCOR are used as part of INRNAV.

5) '(computed)' refers to those quantities computed by 'flight code' as opposed to those generated by 'IMU simulator'.

2.2 Some Conventions

1) Two dimensional arrays, e.g.: A (row, column) are interpreted as

$$\begin{bmatrix} A(1,1) & A(1,2) & A(1,3) \\ A(2,1) & A(2,2) & A(2,3) \\ A(3,1) & A(3,2) & A(3,3) \end{bmatrix}$$

2) Relationship among Rates

A) DELTA * ITRNAV = DELT
ITRNAV = n*ITRATT for some integer n
DELTS * ITRATT = DELTA
ITRFIL = j * ITRNAV for some integer j
ITRATT = k * ITRFIL for some integer k

B) DELTS is found by examining trajectory input tape
ITRATT is an input, then DELTA is computed, ITRATT recomputed
ITRNAV is an input, then DELT is computed, ITRNAV recomputed
INCOR is an input

3) Cn$m (3,3) is transformation matrix from m to n frame

4) Q2P$5(4) is the quaternion representation of the rotation involved in the transformation from the 5 frame to the 2P frame.

Q2P$5(I) - scalar element
Q2P$5(2) - $\hat{x}$ component
Q2P$5(3) - $\hat{y}$ component
Q2P$5(4) - $\hat{z}$ component

5) vector (3) is a vector, stored as
vector (1) - $\hat{x}$ component
vector (2) - $\hat{y}$ component
vector (3) - $\hat{z}$ component

Figure 4. Program Symbol Listing -
Variables - IMU Model Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
RESID (1) thru (3)	DELT	CMINTG	ft/sec, 2I	Residuals after quantization of DV$P, used during following cycle for next quantization.
DV$P(1) thru (3)	DELT	CMINTG	ft/sec, 2I	"Integral" of SF$T over the nav cycle (for SD see HSINT2)
ETA$P(1) thru (4)	DELTA	XXATUD	radians, 2I	"Gimbal angles"; that is, 4 gimbal transformation from 2I to 5 (LL&SS). Incremental angles from gyros (SD only).
WT (1) thru (3)	DELTS	LLINTG	radians/sec, 2I	Inertial angular rate of LLWA frame (LL only).
SF$TP (1) thru (3)	DELTS	XXINTG	ft/sec^2, 2I	Previous value of SF$T, that is, last previous value generated by INREC, as transformed by XXINTG.
C2P$5 (1) thru (3)	DELTA	XXATT	none, 2P	Transformation DCM from 5 frame to 2P frame (computed) at end of cycle.
C$MIS (1) thru (3)	DELTS	LLINTG	none, 2I	Misalignment DCM, that is, transformation from 2P to 2I, at end of cycle (LL only).
C10$0 (1,1) thru (3,3)	At Initialization	SSINIT	none, 10	DCM from 0 to 10, which is used by 'IMU simulator' portion only (SS only).
C10$2P(1,1) thru (3,3)	DELTS	SSINTG	none, 10	DCM from 2P to 10, which is used by 'IMU simulator' portion only (SS only).

Figure 4. Program Symbol Listing -
Variables - IMU Model Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
C2P$5T (1,1) thru (3,3)	DELTS	SDINTG	none, 2P	DCM from 5 to 2P, which is used by 'IMU simulator' portion only (SD only,.
TRESID (1) thru (3)	DELTS	LLINTG	radians, 2I	Residual of torquing angle after quantization (LL only).
ARESID (1) thru (3)	DELTA	SDATUD	radians, 2I	(SD only) Residual of incremental gyro angles after quantization.

Figure 4. Program Symbol Listing -
Variables - Inputs Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
AQUANT	At Initialization	Default or NAMELIST	radians	Attitude Quantization (applied to ETA$P by XXATUD).
TQUANT	"	"	radians	Torquing Angle Quatization (applied to THTTRQ by LLINTG) (LL only).
VQUANT	"	"	ft/sec	Velocity Quantization (applied to DV$P by CMINTG).
START	"	"	sec.	Simulation start time, with reference to times associated with trajectory inputs.
STOP	"	"	sec.	Simulation stop time, with reference to times associated with trajectory inputs.
PLOTIM	"	"	sec,	Plot data output period.
PRNT	"	"	sec,	Print data output period.
RSTRT	"	"	sec,	Restart data output period.
CVD1	"	"	sec^{-1}	Vertical Damping Coefficient.

Figure 4. Program Symbol Listing - Variables - Inputs Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
CVD2	At Initialization	Default or NAMELIST	sec^{-2}	Vertical damping coefficient.
CVD3	"	"	sec^{-3}	Vertical Damping Coefficient
ITRNAV	"	"	none,	Input as navigation computations rate in hertz, changed to a multiple of DELTA by INREC.
INCOR	"	"	none,	COP$2P orthonormalize rate, expressed as multiple of ITRNAV.
IMUTYP	"	"	none,	IMU mechanization 1=>LLWA 2=>SS 3=>SO
IERLIM	"	"	none,	Error limit: Error stop occurs if ISTOP $\geq$ IERLIM
TOLJRK	"	"	ft/sec^3,	Jerk (third time derivative of distance) tolerance as implemented in CMINTG - See CMINTG documentation.
HSOTIM	"	"	sec,	(SD only) Next time (wrt PROFGEN times) to normalize Q2P$5 (or orthonormalize CP$5) as used by HSINTG.
HSOINC	"	"	sec,	(SD only) Time between normalizations as described for HSOTIM.

Figure 4. Program Symbol Listing -
Variables - Inputs Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
IPC(30)	At Initiali-zation	Default or NAMELIST	none,	If 1 compute RHO using exact formula for ellipsoid earth. If 0, use approximation.
IPC (29)	"	"	none,	1=> higher order algorithms (extrapolations and interpolations.
IPC (28)	"	"	none,	1=> use second order DCM update in nav calculations.
IPC (27)	"	"	none,	(SD only) 0=> Use DCM implementation of high-speed calculations 1=> use quaternion.
IPC (26)	"	"	none,	(SD only) 0=> Use 1st order Q2P$5 (C2P$5) update 1=> Use 2nd order 2=> Use 3rd order.
IPC (25)	"	"	none,	1=> Run attitude predictor/filter
IPC (20) thru (24)	"	"	none,	Unused
IPC (1) thru (19)	"	"	none,	Array of switches that control printout of their associated variables 0=> no printout ≠0=> value is printed. - See "IPC List"
IOPNLP	"	"	none,	(LL only) 0=> calculate misalignment matrix C$MIS ≠0=> C$MIS stays = identity matrix

Figure 4. Program Symbol Listing -
Variables - Inputs Common BLock (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
SSX0	At Initial-zation	Default or NAMELIST	radians,	(SS only) X component of three Euler angles representing initial inertial platform alignment C0$10.
SSY0	"	"	radians,	(SS only) Y component of three Euler angles - See SSX0
SSZ0	"	"	radians,	(SS only) Z component of three Euler angles - See SSX0
RSMAX	"	"	radians,	Maximum value of filter residuals allowed before switching gains column pointer to left
ITRFIL	"	"	none,	Number of attitude cycles per cycle of attitude filter.
DRSMAX	" ,	"	radians/sec,	Maximum value of change in filter residuals allowed before switching gains column pointer to left

Figure 4. Program Symbol Listing -
Variables - Control Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
OTIME	DELT	INRNAV	seconds,	Next time to orthonormalize C0P$2P
PTIME	DELT	EXEC	seconds,	Next time to print.
PLTIME	DELT	EXEC	seconds,	Next time to plot.
RSTIME	DELT	EXEC	seconds,	Next time to output restart data.
INIT	DELT	EXEC	none,	pass counter: zero → initialization; one → first pass; two → subsequent pass
TPREV	DELTS	EXEC	seconds,	Previous time (time at end of previous sample cycle).
ISTOP	Various times	Various routines	none,	Severity level of error; used by EXEC.
ICMCYL	At Initialization	XXINIT	none,	Number of DELTS cycles over which CMINTG is integrating.
ICMCYP	DELTS	CMINTG	none,	Number of cycle (1 thru ICMCYL) that CMINTG will next process.

Figure 4. Program Symbol Listing -
Variables - Trajectory Input Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
SLAT$T	DELTS	INREC	none,	Sine of LAT$T
CLAT$T	"	"	none,	Cosine of LAT$T
SALF$T	"	"	none,	Sine of ALF$T
CALF$T	"	"	none,	Cosine of ALF$T
DELTS	At Initialization	"	sec,	Time between PROFGEN inputs, that is, difference between TIME$ on successive records.
TIME	DELTS	INREC	seconds,	Time at end of cycle, read ITRATT times per attitude cycle.
LAT$T	"	"	radians, OP	Geodetic Latitude, at end of cycle.
LONG$T	"	"	radians, OP	Geodetic Longitude, at end of cycle.
ALF$T	"	"	radians, OP	Alpha angle (angle of rotation, in azimuth, of 2P frame from 2 frame), at end of cycle.

Figure 4. Program Symbol Listing -
Variables - Trajectory Input Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
HB	DELTS	INREC	feet, 2P	Geodetic Altitude, at end of cycle.
ETA$T (1) thru (3)	"	"	radians, 2P	Roll (X-rot), Pitch (Y-rot), and Yaw (z-rot). Transformation from 5 frame to 2P frame; at end of cycle. (Actually 5Q to 2QP, but 5 to 2P is identical).
V$T (1) thru (3)	"	"	feet/sec, 2QP	'Groundspeed" (instantaneous velocity,with respect to the earth-fixed point 'beneath' the vehicle); at end of the cycle.
SF$T (1) thru (3)	"	"	feet/sec^2,2QP	Specific force sensed at end of cycle [transformed to 2I frame by XXINTG]
TIME0	At Initiali-zation	"	seconds,	Starting time.
LONG0	"	"	radians, OP	Starting geodetic longitude.

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
C0P$2P (1,1) thru (3,3)	DELT	INRNAV	none, 0P	DCM (computed) from 2P to 0P, at end of cycle.
V$2P (1) thru (3)	"	"	feet/sec, 2P	Ground speed (see V$T for explanation) at end of cycle (computed)
DV$2P (1) thru (3)	"	"	feet/sec, 2P	See DV$P - delta V as used by INRNAV.
RHO (1) thru (3)	"	"	radians/sec,2P	Angular rate of craft with respect to earth fixed frame, around level axes only (computed).
C$RHO(1,1) thru (3,3)	"	"	radians/sec,2P	Skew-symmetric matrix formed from RHO*DELT vector.
A (1,1) thru (3,3)	-	-	-	Scratch Matrix
B (1,1) thru (3,3)	-	-	-	Scratch Matrix
THTERT(1) thru (3)	DELT	INRNAV	radians, 0P	Rotation of earth, wrt inertial space, in DELT time (computed).
THTTRQ (1) thru (3)	"	"	radians, 0P	Rotation of 2P frame, wrt inertial space, needed to keep 2P frame level and "wander azimuth" (computed)

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
THTCOR (1) thru (3)	DELT	INRNAV	radians, 0P	Term used in computation of WXV term (computed).
V$2 (1) thru (3)	DELT	POSVEL	ft/sec, 2	Groundspeed in 2-frame (Up-East-North) (computed).
DELT	At Initiali-zation	INREC	sec,	Navigation Computations cycle period.
FILATT (1,1) thru (3,3)	DELTA *ITRFIL	ATTFIL	radians radians/sec radians/sec^{2}2P	State vector used by attitude filter; filtered attitude, first derivative and second derivative.
DELTA	At Initiali-zation	INREC	sec,	Attitute computations cycle period.
XSALF	DELT	POSVEL	none,	Sine of Alpha angle (computed) extrapolated to middle of next cycle.
XCALF	DELT	POSVEL	none,	Cosine of Alpha angle (computed) extrapolated to middle of next cycle.
XOP$2P (1,1) thru (3,3)	DELT	INRNAV	none, 0P	DCM (computed) from 2P to 0P, as extrapolated to middle of next nav cycle.
HV$2P(1) thru (3)	"	"	ft/sec, 2P	Groundspeed (see V$T for explanation) at middle of cycle, as interpolated (computed).

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
OV$2P(1) thru (3)	DELT	INRNAV	ft/sec, 2P	Groundspeed (see V$T for explanation) at beginning of cycle (computed) as V$2P from last cycle.
S2PHI	DELT	GRAV	none, OP	Square of sine of lattitude, at end of cycle (computed).
WXV (1) thru (3)	DELT	INRNAV	ft/sec, 2P	($\underline{\omega}$ x $\underline{v}$)*DELT form of velocity update equations (computed).
G$2P (1) thru (3)	"	GRAV	ft/sec^2,2P	Gravity vector (computed).
H	"	INRNAV	ft, 2P	Geodetic height (computed).
ALPHA	"	POSVEL	radians,	Alpha angle (computed), at end of cycle.
SALF	"	"	none,	Sine of ALPHA
CALF	"	"	none,	Cosine of ALPHA.
ETA (1) thru (3)	"	"	radians, 2	Roll, pitch, and heading transformation from 5 to 2 frame (computed).

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
OHB	DELT	INRNAV	ft, 2P	Previous value of HB.
CO$5 (1,1) thru (3,3)	DELTA	SDATUD	none, O	(SD only) DCM from 5 frame to O frame, used for generating incremental attitude angles.
DV5$2P(1) thru (3)	DELTA	HSINTG	ft/sec, 2P	(SD only) values of ΔV, in 2P frame, accumulated over a nav cycle.
ODX	DELTA	HSINTG	radians, 2P	(SD only) previous value of ΔX rotation of 5-frame wrt 2P-frame.
ODY	DELTA	HSINTG	radians,2P	(SD only) previous value of Δy rotation - see ODX.
ODZ	DELTA	HSINTG	radians, 2P	(SD only) previous value of Δz rotation - see ODX.
Q2P$5(1) thru (4)	DELTA	HSINTG	none, 2P	Quaternion representation of rotation implicit in transformation from 5-frame to 2P-frame.
GAINS (1,1) thru (3,10)	At Initiali-zation	Default or NAMELIST	none,	10 columns of fixed gains for attitude filter.
OFILTR(1) thru (3)	DELTA* ITRFIL	ATTFIL	radians,	Old filter residuals, for use with next cycle of of attitude filter

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
VDMP	DELT	INRNAV	ft/sec, 2P	Vertical velocity damping term, to be added in during next comp cycle (computed).
LAT	"	POSVEL	radians, OP	Geodetic Latitutde (computed).
LONG	"	"	radians, OP	Geodetic longitude (computed).
XS2PHI	"	GRAV	none,	Square of sine of latitude, extrapolated to middle of next cycle.
XV$2P (1) thru (3)	"	INRNAV	ft,sec, 2P	Ground ed (see V$T for explanation) extrapolated to midc e of next cycle.
XH	"	"	ft, 2P	Geodetic height, as extrapolated to middle of next cycle.
DELH	"	"	ft, 2P	Difference, at end of cycle, between computed height (H) and barometric height (HB) (computed).
CO$10 (1,1) thru (3,3)	At Initali-zation	SSINIT	none, 0	(SS only) DCM from 10 frame to O frame.
COP$O (1,1) thru (3,3)	DELT	SSTFRM	none, OP	(SS only) DCM from O frame to OP frame.

Figure 4. Program Symbol Listing -
Variables - Navigation Calculations Common Block (Cont.)

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
IGAINP(1) thru (3)	DELTA* ITRFIL	ATTFIL	none,	Column of gains matrix last used for update of yaw, pitch and roll, respectively.

Figure 4. Program Symbol Listing -
Variables - "Precision" Common Block

Name	Generated Every	By	Expressed in Units, Frame	Description/Comments
SQRPTH	At Initialization	Default or NAMELIST	none,	Value of 1,2,3, or 4 specifies algorithm to use in computing square root.
ATNPTH	"	"	none,	Value of 1,2,3, or 4 specifies algorithm to use in computing arc tanget.
SINPTH	"	"	none,	Value of 1,2,3 or 4 specifies algorithm to use in computing sine.
COSPTH	"	"	none,	Value of 1,2,3 or 4 specifies algorithm to use in computing cosine.
FRACTL	"	"	bits	Value from 16 to 48 (53 on IBM/360) specifies length of 'simulated' fraction.
EXPOL	"	"	bits	Value from 5 to 8 specifies length of 'simulated' exponent.
ROUND	"	"	none	1 imples arithmetic operations are rounded, 0 implies truncation.

Variables that can be printed out under control of IPC(1) thru (19)

IPC (index)	Variable
1	WT (1) thru (3)
2	ETA$P (1) thru (4)
3	DV$2P (1) thru (3)
4	V$2P (1) thru (3)
5	RHO (1) thru (3)
6	COP$2P (1,1) thru (3,3)
7	-unused-
8	THTERT (1) thru (3)
9	THTTRQ (1) thru (3)
10	WXV (1) thru (3)
11	VDMP
12	G$2P (1) thru (3)
13	elements (1,2), (1,3) and (2,3) of DCM representing incremental rotation of C$RHO
14	C10$2P (1,1) thru (3,3)
15	C2P$5 (1,1) thru (3,3)
16	CO$5 (1,1) thru (3,3)
17	TIME
18	element (3,2), minus (3,1), (2,1) of $(C2P\$5)^{T} \cdot (C2P\$5T)$ {matrix multiply implied}
19	Q2P$5 (1) thru (4)

2.3 DESCRIPTION OF ROUTINES

MAIN

2.3.1 Function & Equations

This is the executive routine for the simulator. It receives control when the simulator is loaded for execution and then calls the appropriate subroutines in the appropriate sequence. This routine also determines when it is time for periodic printout and plot tape output and calls the necessary routines. It is also responsible for reading the operator specified,and PROFGEN generated,initialization parameters. Finally this routine terminates execution when the operator specified stop time is exceeded or errors which exceed the operator specified error limit are detected.

Input Variables

ITITLE - 80 character operator specified title placed on each printout page.

IPAGE - Number of next page to be printed.

IMUTYP - Operator specified code to select the type of IMU to be simulated.

1 = Local Level Wander Azimuth (LLWA).

2 = Space Stable (SS).

3 = Strapdown (SD).

ITRNAV - Number of attitude update iterations per navigation code.

ITRATT - Number of trajectory samples per attitude update cycle.

ISTOP - Error severity indicator.

TIME - Elapsed time for trajectory.

STOP - Operator specified stop time.

TPREV - Time for previous cycle.

SF$T - Specific force from trajectory.

ITRFIL - Number of attitude update iterations per attitude filter cycle.

PTIME - Elapsed time for next printout.

PLTIME - Elapsed time for next plot tape record output.

PRNT - Periodic printout period.

PLOTIM - Plot tape output period.

IOPNLP - Flag to control local level wander azimuth IMU model. "0" value indicates misalignment due to torquing commands is to be modeled. "1" value indicates no misalignment modeled.

SSXO, SSYO, SSZO - X, Y, and Z Euler angles for initial space stable platform alignment.

HSOTIM - Simulated time at which Q2P$5 or C2P$5 is to be normalized.

HSOINC - Time between normalizations for Q2P$5 or C2P$5.

GAINS - Gains for attitude filter.

Internal Variables

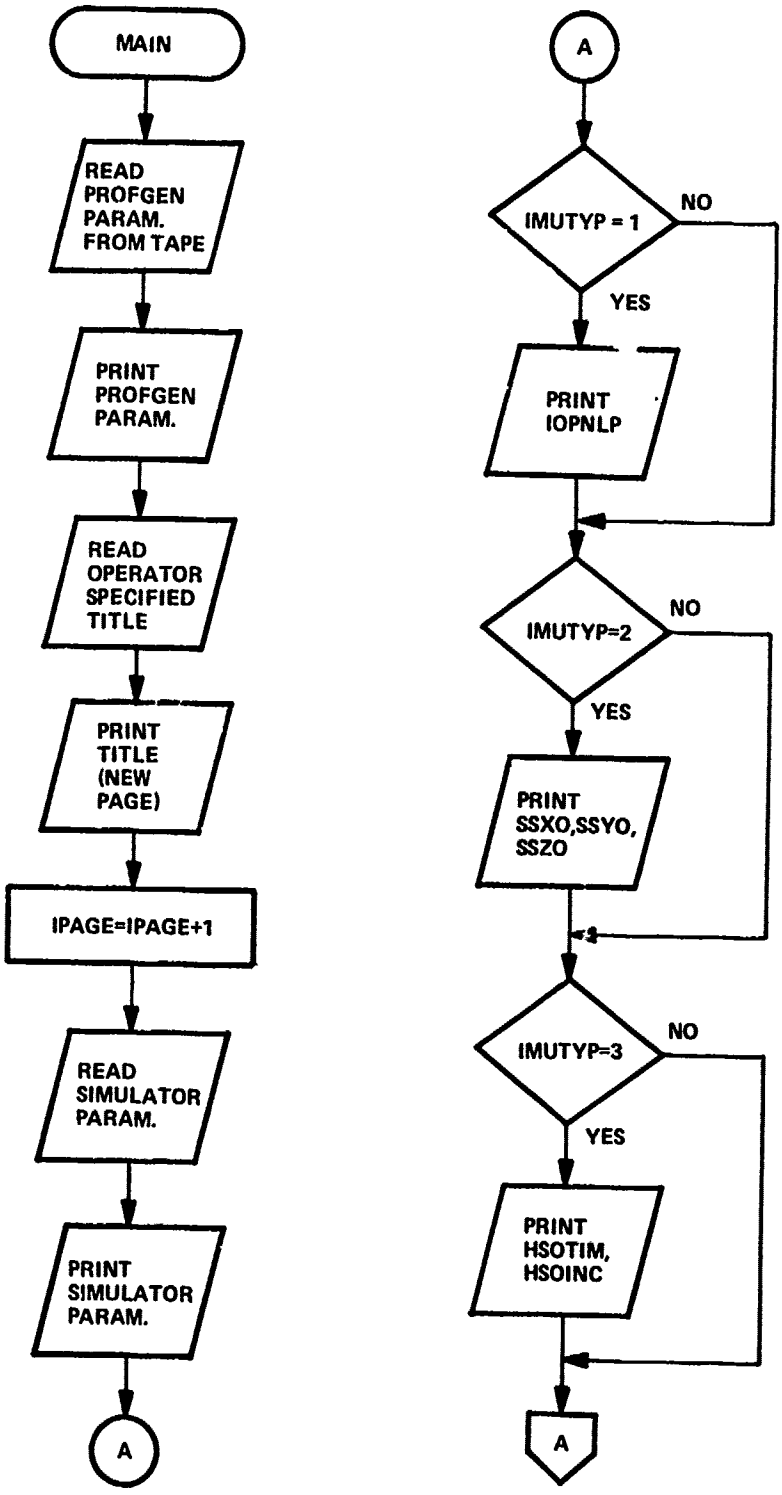
None

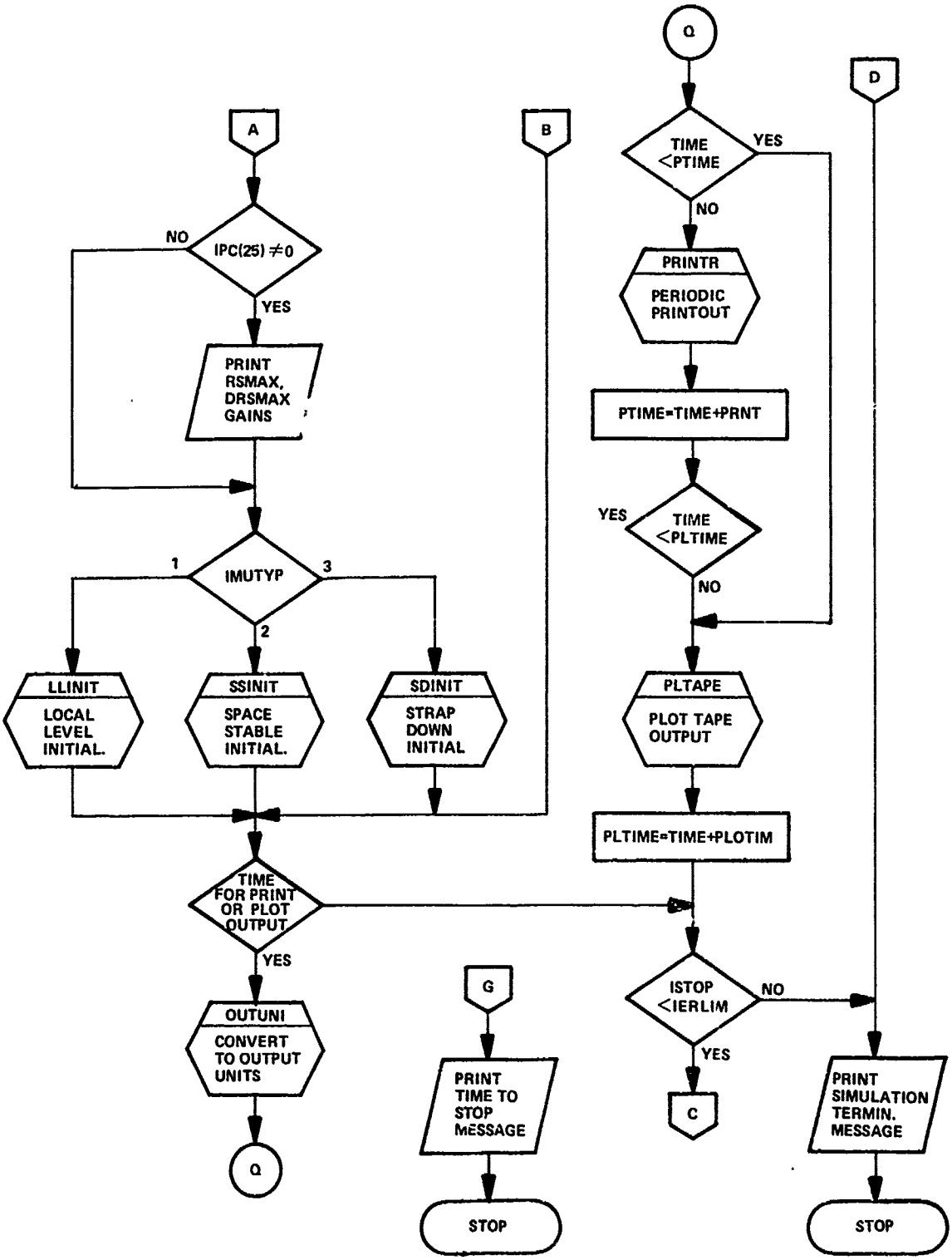
Output Variables

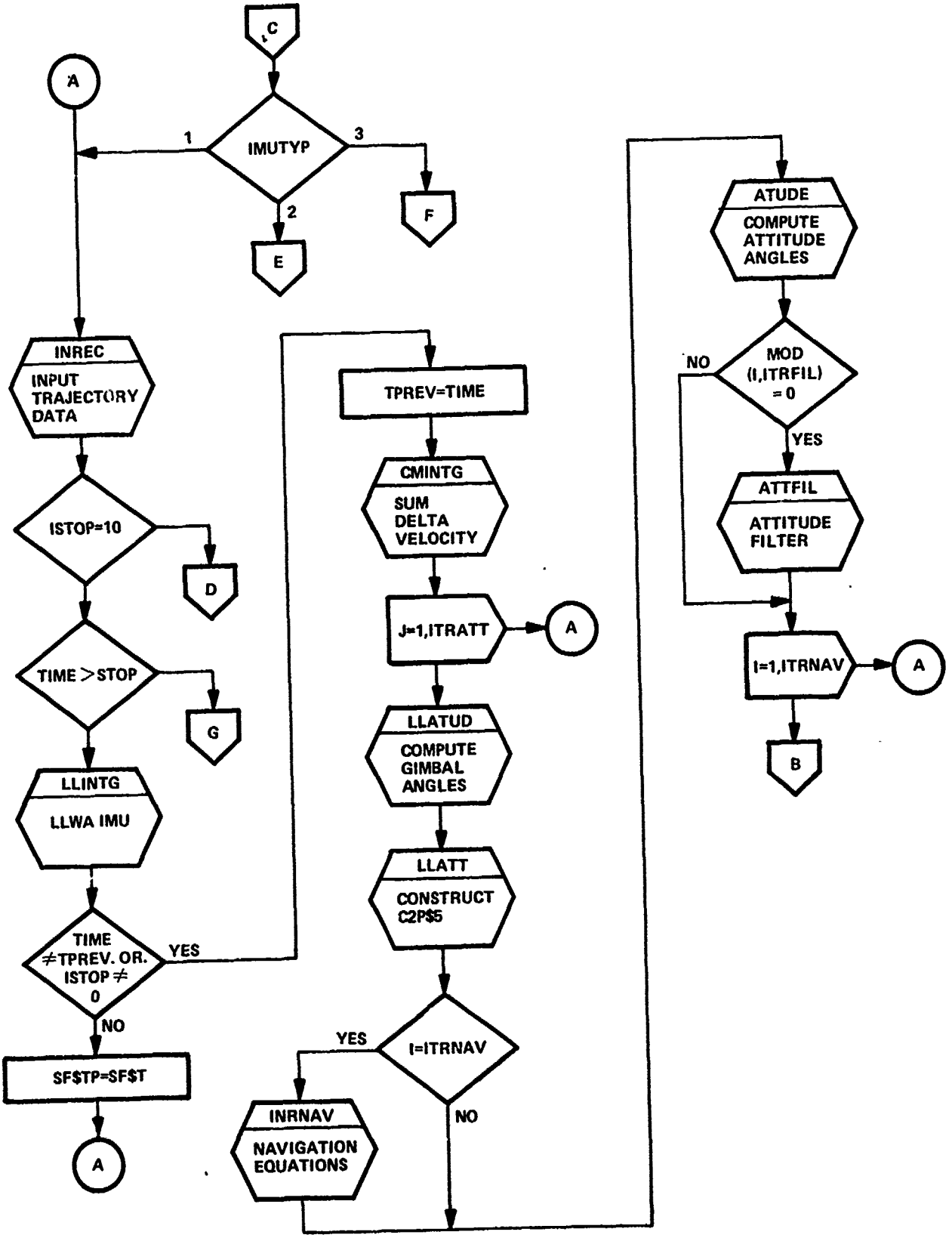
TPREV - See above.

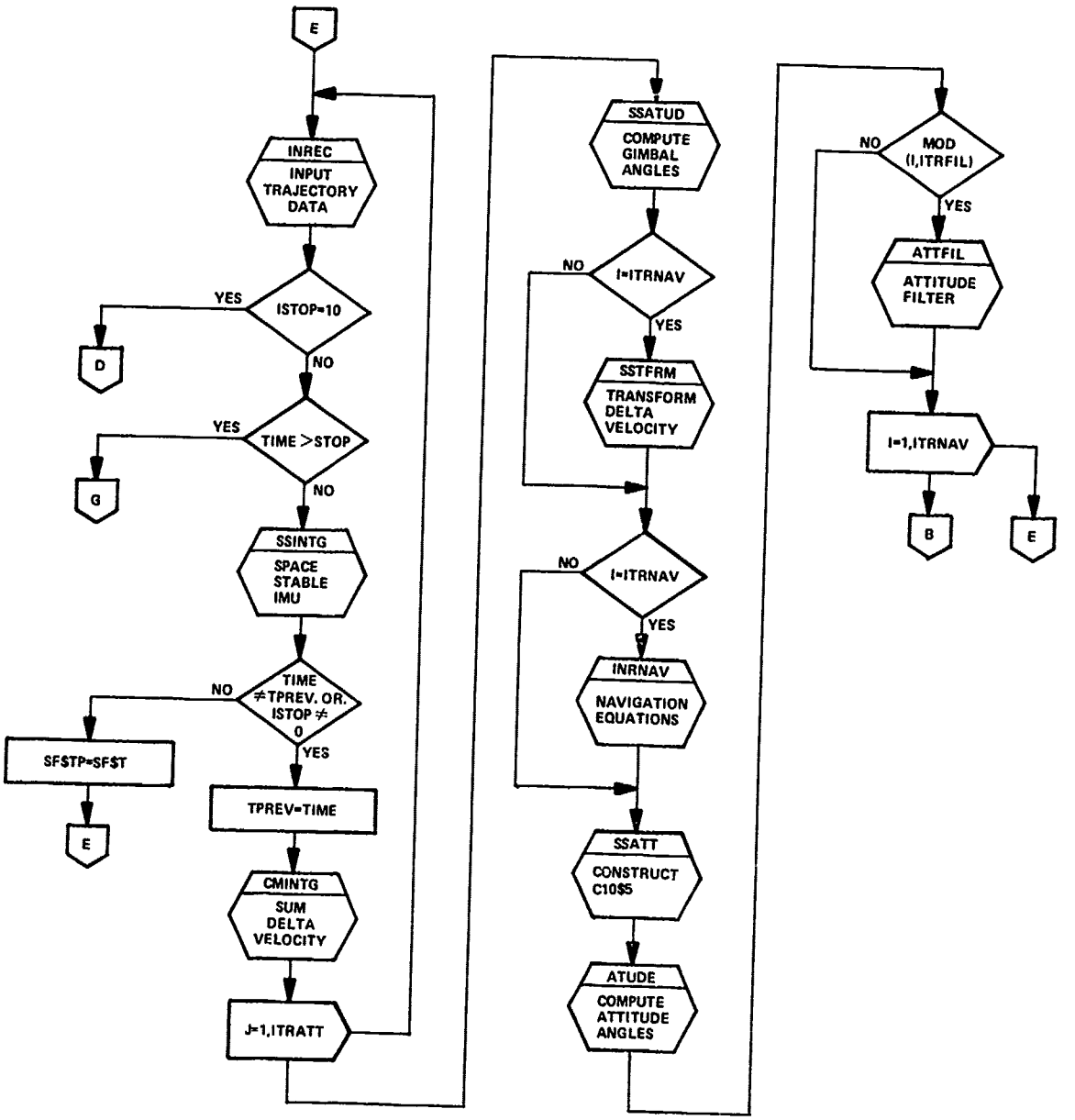
SF$TP - Specific force from previous cycle.

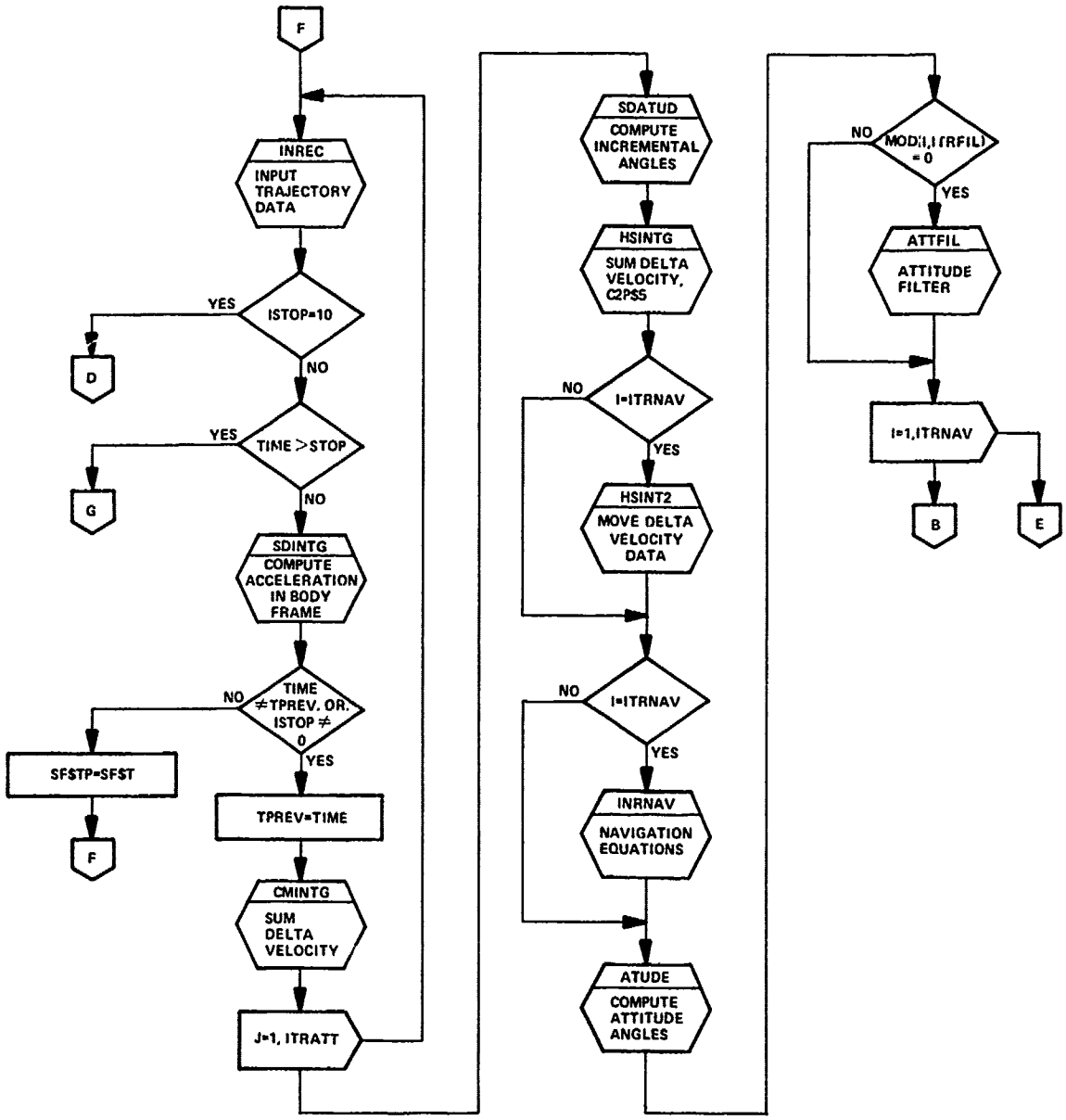
IPAGE - See above.


MAIN
READ PROFGEN PARAM. FROM TAPE
PRINT PROFGEN PARAM.
READ OPERATOR SPECIFIED TITLE
PRINT TITLE (NEW PAGE)
IPAGE=IPAGE+1
READ SIMULATOR PARAM.
PRINT SIMULATOR PARAM.
A
A
IMUTYP = 1
NO
YES
PRINT IOPNLP
IMUTYP=2
NO
YES
PRINT SSXO,SSYO, SSZO
IMUTYP=3
NO
YES
PRINT HSOTIM, HSOINC
A


A
NO
IPC(25) ≠0
YES
PRINT RSMAX, DRSMAX GAINS
1
IMUTYP
3
2
LLINIT
LOCAL LEVEL INITIAL.
SSINIT
SPACE STABLE INITIAL.
SDINIT
STRAP DOWN INITIAL
B
TIME FOR PRINT OR PLOT OUTPUT
YES
OUTUNI
CONVERT TO OUTPUT UNITS
Q
G
PRINT TIME TO STOP MESSAGE
STOP
Q
TIME <PTIME
YES
NO
PRINTR
PERIODIC PRINTOUT
PTIME=TIME+PRNT
YES
TIME <PLTIME
NO
PLTAPE
PLOT TAPE OUTPUT
PLTIME=TIME+PLOTIM
ISTOP <IERLIM
NO
YES
C
D
PRINT SIMULATION TERMIN. MESSAGE
STOP


C
A
IMUTYP
1
2
3
E
F
INREC
INPUT TRAJECTORY DATA
ISTOP=10
D
TIME >STOP
G
LLINTG
LLWA IMU
TIME ≠TPREV. OR. ISTOP ≠ 0
YES
NO
SF$TP=SF$T
A
TPREV=TIME
CMINTG
SUM DELTA VELOCITY
J=1,ITRATT
A
LLATUD
COMPUTE GIMBAL ANGLES
LLATT
CONSTRUCT C2P$5
I=ITRNAV
YES
NO
INRNAV
NAVIGATION EQUATIONS
ATUDE
COMPUTE ATTITUDE ANGLES
MOD (I,ITRFIL) = 0
NO
YES
ATTFIL
ATTITUDE FILTER
I=1,ITRNAV
A
B


E
INREC
INPUT TRAJECTORY DATA
ISTOP=10
YES
D
NO
TIME >STOP
YES
G
NO
SSINTG
SPACE STABLE IMU
TIME ≠TPREV. OR. ISTOP ≠ 0
NO
SF$TP=SF$T
E
YES
TPREV=TIME
CMINTG
SUM DELTA VELOCITY
J=1,ITRATT
SSATUD
COMPUTE GIMBAL ANGLES
I=ITRNAV
NO
YES
SSTFRM
TRANSFORM DELTA VELOCITY
I=ITRNAV
NO
YES
INRNAV
NAVIGATION EQUATIONS
SSATT
CONSTRUCT C10$5
ATUDE
COMPUTE ATTITUDE ANGLES
MOD (I,ITRFIL)
NO
YES
ATTFIL
ATTITUDE FILTER
I=1,ITRNAV
B
E


F
INREC
INPUT TRAJECTORY DATA
ISTOP=10
YES
NO
D
TIME > STOP
YES
NO
G
SDINTG
COMPUTE ACCELERATION IN BODY FRAME
TIME ≠ TPREV. OR. ISTOP ≠ 0
NO
YES
SFSTP=SFST
F
TPREV=TIME
CMINTG
SUM DELTA VELOCITY
J=1, ITRATT
SDATUD
COMPUTE INCREMENTAL ANGLES
HSINTG
SUM DELTA VELOCITY, C2PSS
I=ITRNAV
NO
YES
HSINT2
MOVE DELTA VELOCITY DATA
I=ITRNAV
NO
YES
INRNAV
NAVIGATION EQUATIONS
ATUDE
COMPUTE ATTITUDE ANGLES
MOD(I,ITRFIL) = 0
NO
YES
ATTFIL
ATTITUDE FILTER
I=1,ITRNAV
B
E

BLOCK DATA SUBPROGRAM

2.3.2 Function & Equations

The sole purpose of this routine is to perform initialization of the variables in the various labelled common blocks. As such, it does not consist of any executable code, but merely of DATA statements.

Inputs, Outputs of Internal Variables

Refer to listing.

2.3.3 Function & Equations

The purpose of INREC is to read PROFGEN data from an input tape. Input PROFGEN records are expected to be fixed length (14 words per record) and contain the following variables in single precision floating point:

- TIME - Elapsed time in seconds.
- LAT$T - Geodetic latitude (RAD).
- LONG$T - Geodetic longitude (RAD).
- ALF$T - Wander angle (RAD).
- HB - Altitude (FT).
- ETA$T - Roll, pitch, and yaw atitude angles (RAD).
- V$T - Instantaneous velocity in local level wander azimuth frame (north, west, up axes) (FT/SEC).
- SF$T - Specific force in local level wander azimuth frame (FT/SEC^2).

The routine also provides the sine and cosine of both LAT$T and ALF$T.

The first pass through the routine is used for initialization during which the following steps are taken:

1. Find the operator specified start time on the trajectory tape. If unable to find start time, inform operator with a printed message.
2. Ensure that the attitude update period, the navigation equation update period, and the attitude filter update period are multiples of the trajectory tape period. Also ensure that the navigation period and the attitude filter period are multiples of the attitude period.
3. Printout the timing relations between trajectory, navigation, and attitude periods. Also printout messages to indicate any error conditions.

The following actions are taken when error conditions are detected:

1. Attitude update period is set to the trajectory sample period if the operator specified period is not an even multiple of the trajectory period.
2. Navigation equation period is set to the attitude update period if the operator specified period is not an even multiple of the attitude update period.
3. Attitude filter period is set to the attitude update period if the operator specified period is not an even multiple of the attitude update period.

Input Variables

INIT - Initialization flag. "0" value indicates first pass through routine.

ITRATT - Operator specified attitude update. Rate in cycles per second. Value less than or equal to zero forces rate to that of trajectory samples.

ITRNAV - Operator specified navigation equation. Iteration rate in cycles per second. Value less than or equal to zero forces rate to the rate of attitude computations.

ITRFIL - Operator specified attitude filter. Iteration rate in cycles per second. Value less than or equal to zero forces rate to the rate of attitude computations.

IPC(25) - Attitude filter flag. A non zero value specified by operator indicates attitude filtering is to be performed.

START - Operator specified start time in elapsed seconds.

Internal Variables

PRDAT - Trajectory data input buffer. Equivalenced to trajectory variables.

PRD - Temporary buffer for second trajectory data record. Used in order to determine trajectory sample period.

TIM - Elapsed time in seconds of second trajectory data record.

ITMP1
ITMP2
ITMP3
ITMP4
ITMP5
TEMP

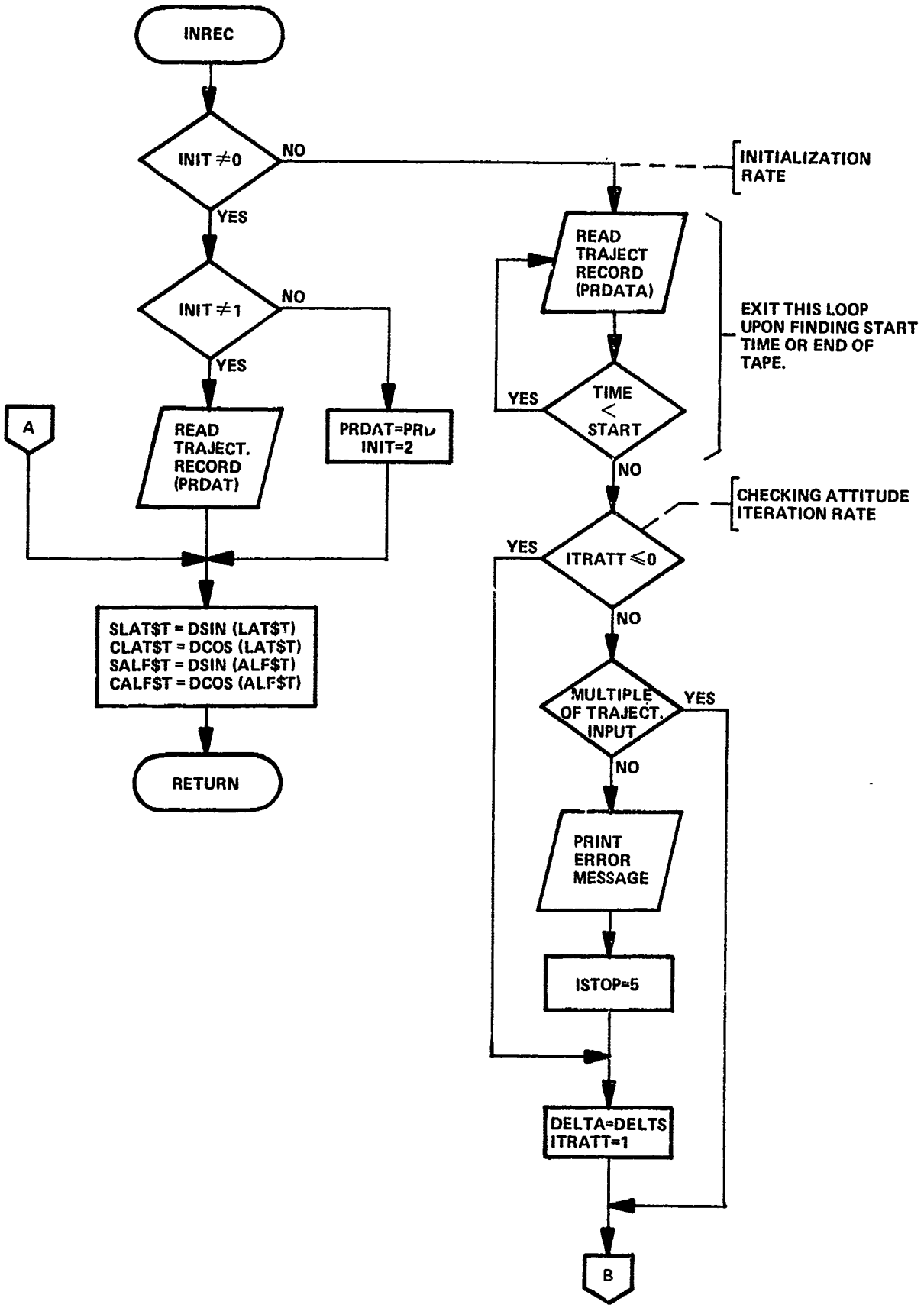
Scratch variables used to calculate timing multiples and for printout.

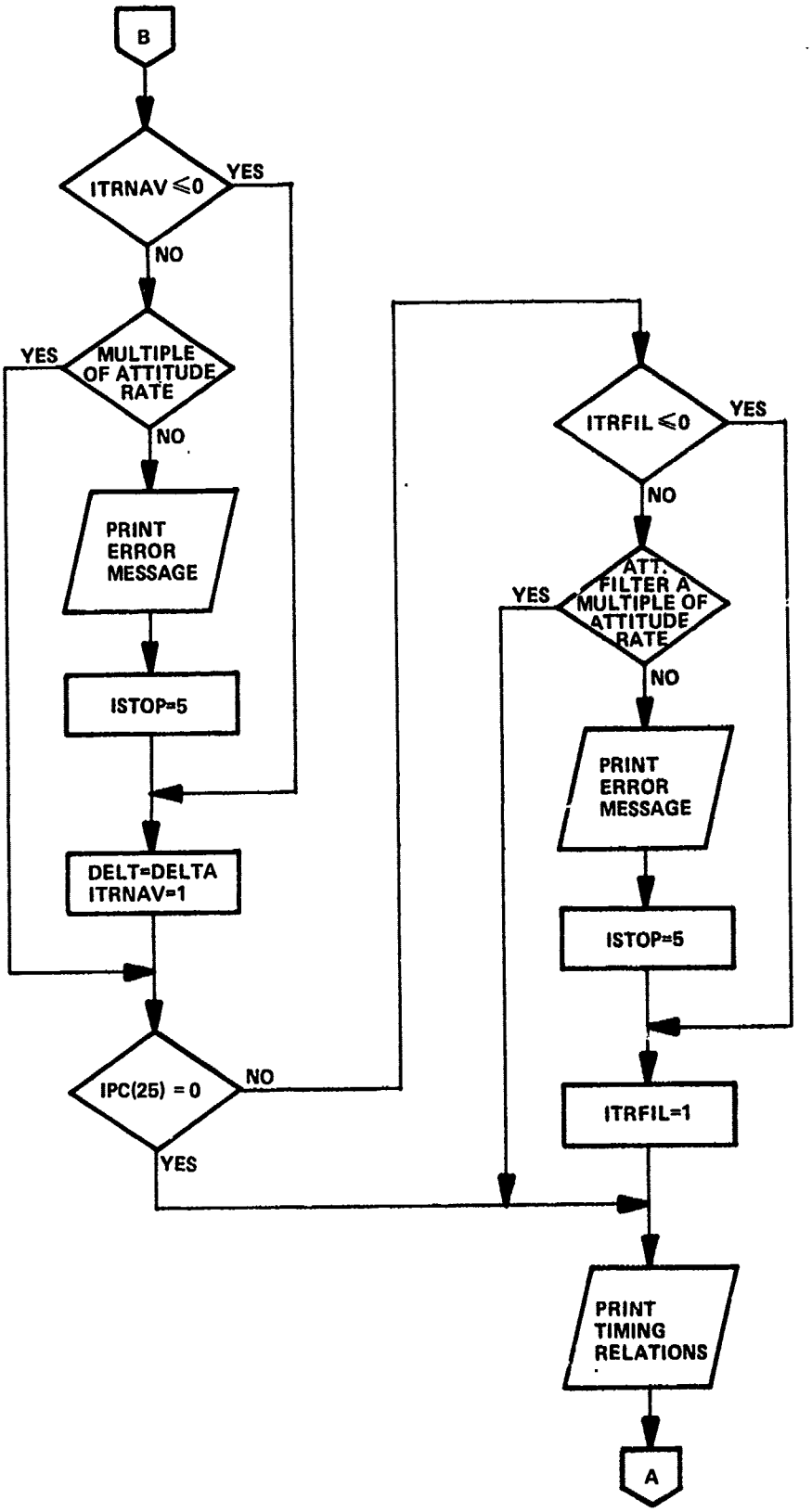
Output Variables

TIME
LAT$T
LONG$T
ALF$T
ETA$T
V$T
SF$T

Data from trajectory tape (see above).

SLAT$T - Sine of LAT$T.
CLAT$T - Cosine of LAT$T.
SALF$T - Sine of ALF$T.
CALF$T - Cosine of ALF$T.
DELTS - Trajectory sample period.
DELTA - Attitude computations iteration period.
DELT - Navigation equations iteration period.
ITRATT - Number of trajectory samples per attitude computation iteration.
ITRNAV - Number of attitude computation iterations per navigation cycle.
ITRFIL - Number of attitude computation iterations per attitude filter cycle.
ISTOP - Error severity level.


INREC
INIT ≠0
NO
YES
INITIALIZATION RATE
INIT ≠1
NO
YES
A
READ TRAJECT. RECORD (PRDAT)
PRDAT=PRV INIT=2
READ TRAJECT RECORD (PRDATA)
EXIT THIS LOOP UPON FINDING START TIME OR END OF TAPE.
YES
TIME < START
NO
CHECKING ATTITUDE ITERATION RATE
YES
ITRATT ≤0
NO
SLAT$T = DSIN (LAT$T)
CLAT$T = DCOS (LAT$T)
SALF$T = DSIN (ALF$T)
CALF$T = DCOS (ALF$T)
MULTIPLE OF TRAJECT. INPUT
YES
NO
RETURN
PRINT ERROR MESSAGE
ISTOP=5
DELTA=DELTS ITRATT=1
B


B
ITRNAV ≤0
YES
NO
YES
MULTIPLE OF ATTITUDE RATE
NO
PRINT ERROR MESSAGE
ISTOP=5
DELT=DELTA
ITRNAV=1
IPC(25) = 0
NO
YES
ITRFIL ≤0
YES
NO
YES
ATT. FILTER A MULTIPLE OF ATTITUDE RATE
NO
PRINT ERROR MESSAGE
ISTOP=5
ITRFIL=1
PRINT TIMING RELATIONS
A

2.3.4 Function & Equations

A) The purpose of TORCOR is to implement the portion of the navigation equation that computes

a) The angular rate of the earth with respect to inertial space $\underline{\Omega}$ (or, after being multiplied by DELT, THTERT).

b) The angular rate of the platform with respect to the inertial space $(\rho + \Omega)$ (or, after being multiplied by DELT, THTTRQ).

c) The $(\underline{w} \times \underline{v}) \times \Delta t$ term of the velocity update (THTTRQ + THTERT) x AV$2P), where x is vector cross product operation and AV$2P is the velocity passed to the subroutine.

B) THTERT is computed as

$$\text{DELT} * (\text{A0P\$2P})^{T} * \begin{bmatrix} 0. \\ 0. \\ \text{WERT} \end{bmatrix}$$

where WERT is the rate of the earth in the North - East - Up frame

The axes orientation of the inertial OP frame is:

X_o = Equatorial - initial meridian

Y_o = Equatorial - initial east

Z_o = Polar - north polar

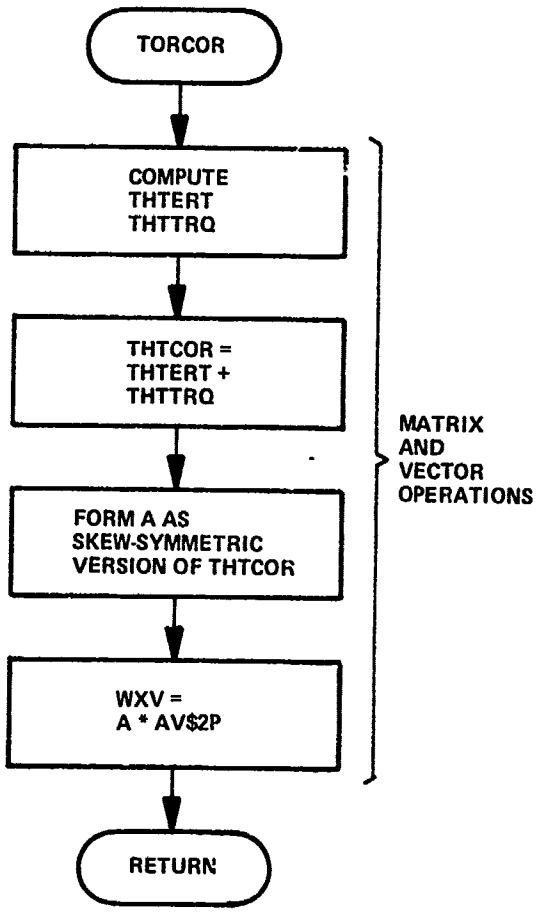
A0P$2P represents the DCM (from 2P to 0P) passed to TORCOR. Note that only the bottow row of A0P$2P is used. THTTRQ is computed as

THTERT + RHO * DELT

where RHO is the vector (computed externally) giving the rate of the 2P frame with respect to the OP frame.

Inputs, Outputs and Internal Variables

AV$2P	- parameter representing velocity as seen in 2P (input)
AOP$2P	- Paramater represents DCM transforming 2P to 0P (input)
RHO	- rate of 2P w.r.t. 0P (input)
WERT	- ≈15°/hr (rotational rate of earth) (input)
DELT	- time between navigation cycles (input)
THTERT	- described above (output)
THTTRQ	- described above (output)
THTCOR	- THTTRQ + THTERT (output)
A	- scratch
WXV	- described above (output)


TORCOR
COMPUTE
THTERT
THTTRQ
THTCOR =
THTERT +
THTTRQ
FORM A AS
SKEW-SYMMETRIC
VERSION OF THTCOR
WXV =
A * AV$2P
RETURN
MATRIX
AND
VECTOR
OPERATIONS

2.3.5 Function & Equations

A) The purpose of INRNAV is to perform the calculations of position (that is latitude, longitude and altitude) and "ground-velocity (that is, instantaneous vehicle velocity expressed in the earth fixed frame). The ground velocity, as written in the OP frame is $\dot{\underline{R}}^{op}$ where $\underline{R}^{op}$ is the vector from the center of the earth-fixed frame to the vehicle. We wish to find this vector in the 2P frame; this can be written as $\underline{V}^{2p} = C^{2p}_{op}\ \dot{\underline{R}}^{op}$. Since the physical data relevant to the calculation is accelerometer data, one wishes to find

$$\underline{V}^{2p}(t) = \underline{V}^{2p}(to) + \int_{to}^{t} \dot{\underline{V}}^{2p}\ dt$$

One relates $\dot{\underline{V}}^{2p}$ to the physical data as follows:

$$\dot{\underline{V}}^{2p} = \frac{d}{dt}\ (C^{2p}_{op}\ \dot{\underline{R}}^{op}) = C^{2p}_{op}\ (\underline{\rho}^{op}_{2p,op} \times \dot{\underline{R}}^{op})$$

$$+ C^{2p}_{op}\ \ddot{\underline{R}}^{op} = \underline{\rho}^{2p}_{2p,op} \times \underline{V}^{2p} + C^{2p}_{op}\ \ddot{\underline{R}}^{op}$$

{where $\underline{\rho}^{2p}_{2p,op}$ is the angular rate of the 2P frame wrt 2P frame as expressed in the 2P frame}

Now, let $\underline{R}^{O}$ be position vector in 0 (space stable equatorial Up - East - North) frame.

then $$\dot{\underline{R}}^{O} = \frac{d}{dt}\ C^{O}_{op}\ \underline{R}^{op} = \underline{\Omega}^{O}_{O,op} \times C^{O}_{op}\ \underline{R}^{op} + C^{O}_{op}\ \dot{\underline{R}}^{op}$$

where $\Omega^{O}_{O,op}$ is rate of 0P frame with respect to 0 frame, as expressed in 0 frame. Therefore

$$\ddot{\underline{R}}^{O} = \frac{d}{dt}\ (\underline{\Omega}^{O}_{O,op} \times \underline{R}^{O}) + \frac{d}{dt}\ (C^{O}_{op}\ \dot{\underline{R}}^{op})$$

$$= \dot{\underline{\Omega}}^{o}_{o,op} \times \underline{R}^{o} + \underline{\Omega}^{o}_{o,op} \times \frac{d}{dt} (C^{o}_{op} \underline{R}^{op})$$

$$+ \underline{\Omega}^{o}_{o,op} \times C^{o}_{op} \dot{\underline{R}}^{op} + C^{o}_{op} \ddot{\underline{R}}^{op}$$

{since $\Omega^{o}_{op,o}$ is a constant}

$$= \underline{\Omega}^{o}_{op} \times C^{o}_{op} \dot{\underline{R}}^{op} + \underline{\Omega}^{o}_{o,op} \times \underline{\Omega}^{o}_{o,op} \times \underline{R}^{o}$$

$$+ \underline{\Omega}^{o}_{o,op} \times (C^{o}_{op} \dot{\underline{R}}^{op}) + C^{o}_{op} \ddot{\underline{R}}^{op}$$

Multiplying by C^{2p}_{o}

$$C^{2p}_{o} \ddot{\underline{R}}^{o} = C^{2p}_{op} \ddot{\underline{R}}^{op} - 2 \Omega^{2p}_{op,o} \times \underline{V}^{2p}$$

$$+ C^{2p}_{o} (\underline{\Omega}^{o}_{o,op} \times \Omega^{o}_{o,op} \times \underline{R}^{o})$$

rearranging

$$C^{2p}_{op} \ddot{\underline{R}}^{op} = C^{2p}_{o} \ddot{\underline{R}}^{o} + 2\Omega^{2p}_{op,o} \times \underline{V}^{2p} - C^{2p}_{o} (\underline{\Omega}^{o}_{o,op} \times \underline{\Omega}^{o}_{o,op} \times \underline{R}^{o})$$

Then, by substituting this result into the previous equation for $\dot{\underline{V}}^{2p}$

$$\dot{\underline{V}}^{2p} = \underline{\rho}^{2p}_{2p,op} \times \underline{V}^{2p} + \underline{\Omega}^{2p}_{op,o} \times \underline{V}^{2p} - C^{2p}_{o} (\underline{\Omega}^{o}_{o,op} \times \Omega^{o}_{o,op} \times \underline{R}^{o})$$

$$+ C^{2p}_{o} \ddot{\underline{R}}^{o}$$

The accelerometers measure $\underline{f}^{2p}$, the sum of inertial acceleration $C^{2p}_{o} \ddot{\underline{R}}^{o}$ and gravitational acceleration $\underline{G}^{2p}$. Since the $\Omega \times \Omega \times R$ term is dependent only on position it can be combined with $\underline{G}^{2p}$ to form $\underline{g}^{2p} = \underline{G}^{2p} + C^{2p}_{o} (\underline{\Omega}^{o}_{o,op} \times \underline{\Omega}^{o}_{o,op} \times \underline{R}^{o})$ thereby yielding

$$\underline{V}^{2p} = \underline{f}^{2p} - \underline{g}^{2p} + (\rho^{2p}_{2p,op} + 2\underline{\Omega}^{2p}_{op,o}) \times \underline{V}^{2p}$$

This then yields the velocity difference equation

$$\overset{*}{\underline{V}}^{2p} = \Delta t \left[\underline{f}^{2p} - \underline{g}^{2p} + (\rho^{2p}_{2p,op} + 2\underline{\Omega}^{2p}_{op,o}) \times \underline{V}^{2p}\right] + \text{damp}$$

where $\overset{*}{\underline{V}}^{2p}$ is extrapolated value of $\underline{V}^{2p}$ and 'damp' represents the vertical channel damping terms.

The altitude h is computed as

$$h(t) = h(to) + \int_{to}^{t} V_1 \, dt + \text{damping},$$

where V_1 is the vertical component of $\underline{V}^{2p}$ and 'damping' represents damping terms.

Latitude, longitude and alpha angle are computed from the direction cosine matrix 'DCM' transforming the 2P frame to the OP frame (A rotation of alpha about up, latitude abour east, and longitude change about north). The differential equation for the about change in the DCM is:

$$\dot{C}^{op}_{2p} = C^{op}_{2p} \times \rho^{2p}_{op,2p}$$

where $\rho^{2p}_{op,2p}$ is the angular rate of the 2P frame with respect to the OP frame as coodinatized in the 2P frame. This equation is mechanized, in difference equation form, as a first or second order DCM update. See documentation on ANGVL2 for an explanation of $\rho^{2p}_{op,2p}$.

A third order vertical damping scheme is implemented as follows [the inputs are the ΔV's from the vertical accelerometer and hb, the altitude from the reference (assumed barometric) altimeter]:

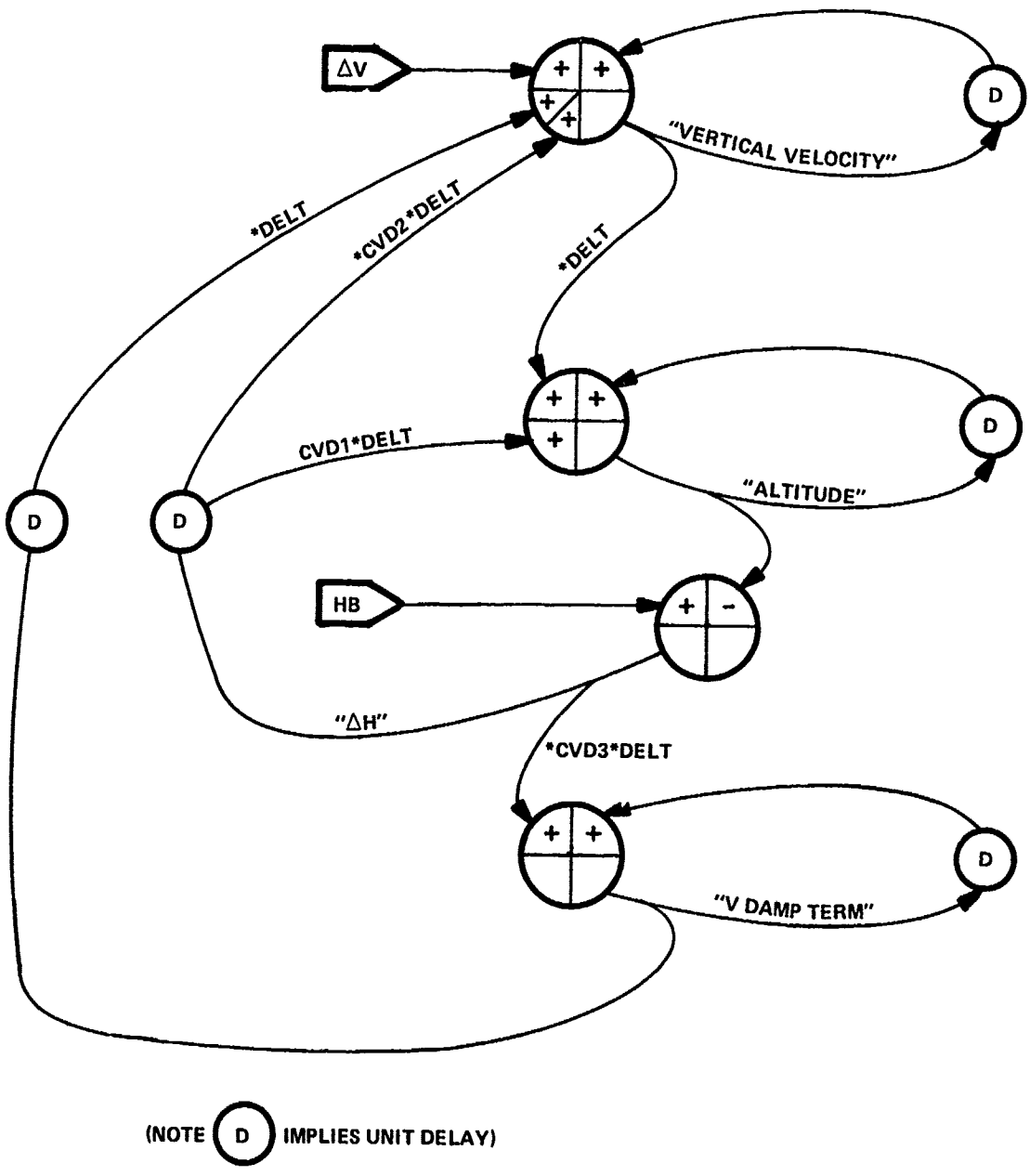


Figure 5. Vertical Damping Flow Diagram

In the above, DELT is the navigation cycle time and CVD1, DVD2 and CVD3 are damping coefficients whose units are sec^{-1}, sec^{-2} and sec^{-3} respectively.

INRNAV also computes THTTRQ, which is the rate of the 2P (computational) frame with respect to the 0 (inertial-initial earth equatorial Up-East-North) frame. This rate is used (external to INRNAV) as follows:

a) Local Level - sent as "torquing commands" to IMU

b) Space stable - unused

c) Strapdown - used to "torque" the 5 (body) frame to 2P (computational) frame DCM.

B) The algorithms used by INRNAV are selected by the values of IPC (28), IPC (29), and IPC (30), which are interpreted as follows:

Variable \ Value	0	1
IPC (28)	Use first order DCM update routine for COP$2P	Use second order DCM update routine for COP$2P
IPC (29)	Use end of current cycle values for V$2P, ALPHA, etc. in computations	Use values of V$2P ALPHA, etc. extrapolated to middle of next cycle, in computations
IPC (30)	Use approximation in computation of angular velocity, ρ	Use exact formula in computation of angular velocity, ρ

C) In broad brush, the calculations of INRNAV are as follows:

1) Update $\underline{v}^{2p}$ (stored as V$2P) using:

a) velocity increments for this navigation cycle ($\underline{f}^{2p}\ \Delta t$) (stored as DV$P)

b) the $(\rho + 2\Omega)$ XV Δt term, as computed on the previous cycle (stored as WXV)

c) the $\underline{g}^{2p}\ \Delta t$ term (computed by multiplying G$2P by DELT)

d) the damping term (computed as (VDMP - CVD2 * DELH) * DELT)

2) If IPC (29) = 1, compute extrapolated value of velocity (stored as XV$2P) and interpolated value of velocity (stored as HV$2P).

3) Compute angular rate of 2P frame with respect to 0P as follows:

a) If IPC (30) = 0
Compute angular velocity using approximate formula (see appendix for derivation)

b) If IPC (30) = 1, then
i) If IPC (29) = 0 compute angular velocity using exact formula with velocity at end of cycle (V$2P)
ii) If IPC (29) = 1 compute angular velocity using exact formula with velocity at middle of current cycle (HV$2P)

4) Update C0P$2P DCM with DCM update routine according to order of update as specified by IPC (28), computing angles from angular velocity. If IPC (29) = 1, form extrapolated DCM X0P$2P. Orthonomalize C0P$2P if it is time to.

5) If IPC (29) = 0, complete calculations as follows:

   a) Use TORCOR to compute torquing angles, WXV terms, based on end-of-cycle values

   b) compute altitude (stored as H) and vertical damping terms DELH, VDMP, based on end-of-cycle values.

   c) Use GRAV to compute gravity, based on end-of-cycle values.

   d) Use POSVEL to compute position and velocity.

6) If IPC (29) = 1, complete calculations as follows:

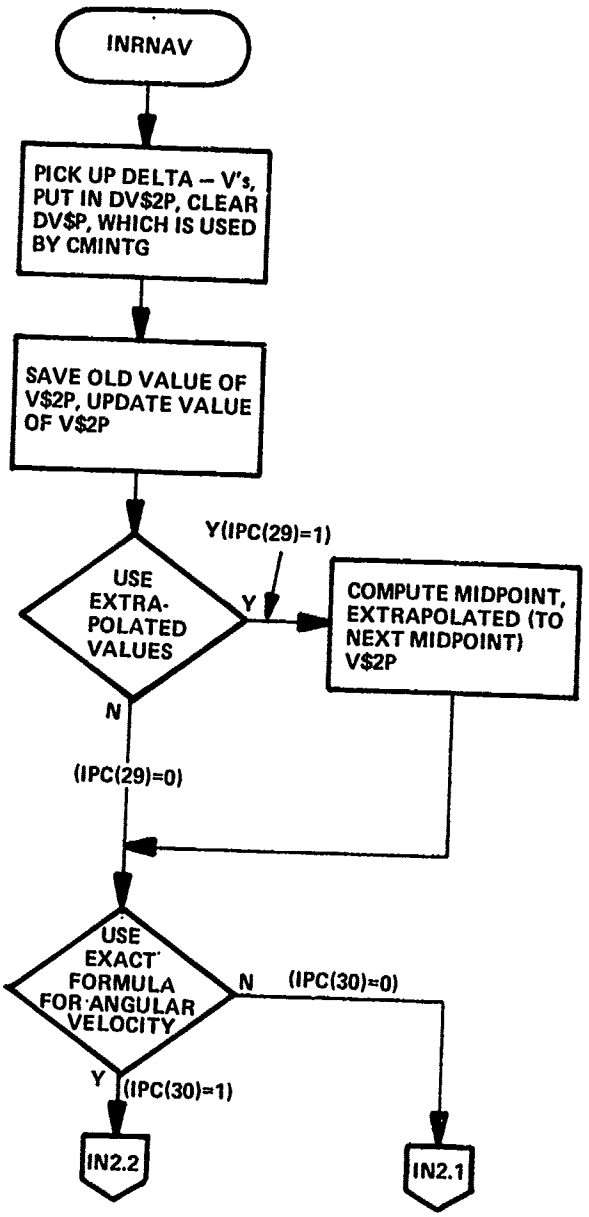
   a) Compute altitude (stored as H), extrapolated altitude, vertical damping terms DELH, VDMP, based on extrapolated values.

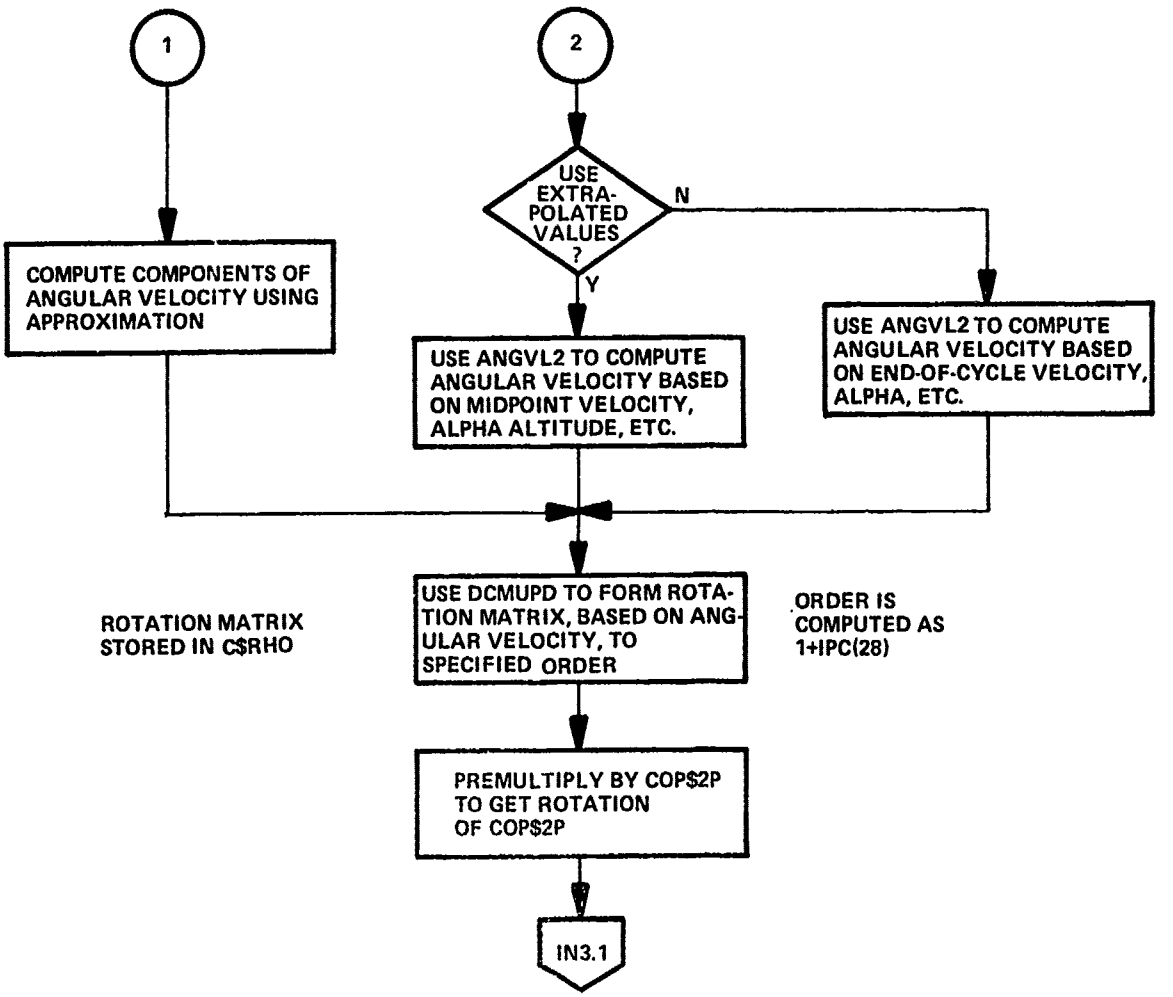
   b) Use GRAV to compute gravity based on extrapolated values.

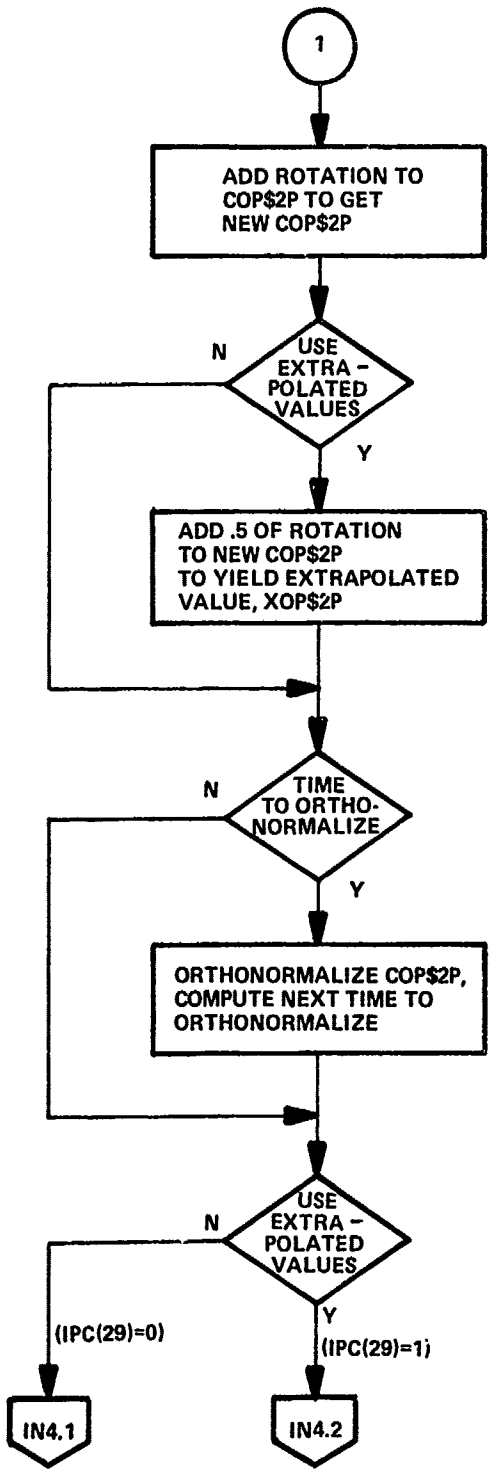
   c) Use POSVEL to compute position and velocity, extrapolated alpha angle.

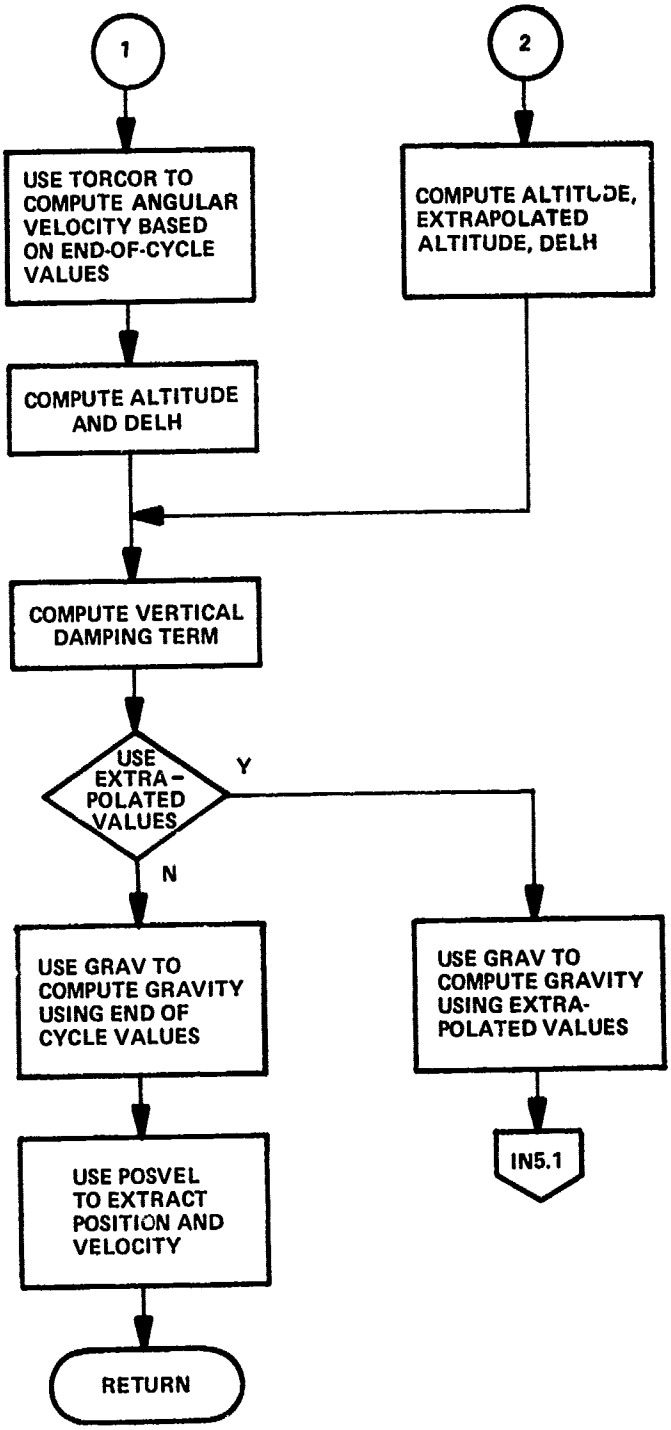
   d) Compute angular velocity using exact formula, based on extrapolated values.

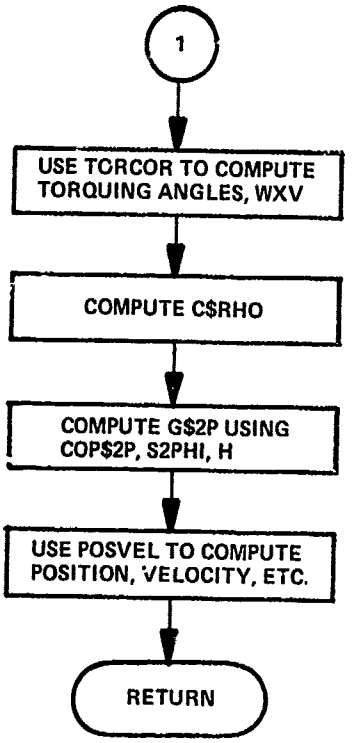
   e) Use TORCOR to compute torquing angles, WXV terms, based on extrapolated values.


INRNAV
PICK UP DELTA – V's, PUT IN DV$2P, CLEAR DV$P, WHICH IS USED BY CMINTG
SAVE OLD VALUE OF V$2P, UPDATE VALUE OF V$2P
Y(IPC(29)=1)
USE EXTRA-POLATED VALUES
Y
COMPUTE MIDPOINT, EXTRAPOLATED (TO NEXT MIDPOINT) V$2P
N
(IPC(29)=0)
USE EXACT FORMULA FOR ANGULAR VELOCITY
N
(IPC(30)=0)
Y
(IPC(30)=1)
IN2.2
IN2.1


1
2
USE EXTRA-POLATED VALUES ?
N
Y
COMPUTE COMPONENTS OF ANGULAR VELOCITY USING APPROXIMATION
USE ANGVL2 TO COMPUTE ANGULAR VELOCITY BASED ON MIDPOINT VELOCITY, ALPHA ALTITUDE, ETC.
USE ANGVL2 TO COMPUTE ANGULAR VELOCITY BASED ON END-OF-CYCLE VELOCITY, ALPHA, ETC.
USE DCMUPD TO FORM ROTATION MATRIX, BASED ON ANGULAR VELOCITY, TO SPECIFIED ORDER
ROTATION MATRIX STORED IN C$RHO
ORDER IS COMPUTED AS 1+IPC(28)
PREMULTIPLY BY COP$2P TO GET ROTATION OF COP$2P
IN3.1


1
ADD ROTATION TO COP$2P TO GET NEW COP$2P
N
USE EXTRA - POLATED VALUES
Y
ADD .5 OF ROTATION TO NEW COP$2P TO YIELD EXTRAPOLATED VALUE, XOP$2P
N
TIME TO ORTHO-NORMALIZE
Y
ORTHONORMALIZE COP$2P, COMPUTE NEXT TIME TO ORTHONORMALIZE
N
USE EXTRA - POLATED VALUES
Y
(IPC(29)=0)
(IPC(29)=1)
IN4.1
IN4.2


1
2
USE TORCOR TO COMPUTE ANGULAR VELOCITY BASED ON END-OF-CYCLE VALUES
COMPUTE ALTITUDE, EXTRAPOLATED ALTITUDE, DELH
COMPUTE ALTITUDE AND DELH
COMPUTE VERTICAL DAMPING TERM
USE EXTRA-POLATED VALUES
Y
N
USE GRAV TO COMPUTE GRAVITY USING END OF CYCLE VALUES
USE GRAV TO COMPUTE GRAVITY USING EXTRA-POLATED VALUES
IN5.1
USE POSVEL TO EXTRACT POSITION AND VELOCITY
RETURN


1
USE TORCOR TO COMPUTE TORQUING ANGLES, WXV
COMPUTE C$RHO
COMPUTE G$2P USING COP$2P, S2PHI, H
USE POSVEL TO COMPUTE POSITION, VELOCITY, ETC.
RETURN

2.3.6 Function & Equations

A) The purposes of this subroutine is to

a) calculate longitude, latitude, and alpha angle

b) compute sine and cosine of alpha, and extrapolate values of the sine and cosine of alpha

c) compute velocity in the 2 frame (Up-East-North)

B) The C0P$2P matrix is, analytically:

$c\phi cl$	$s\phi s\alpha cl - c\alpha sl$	$-c\alpha s\phi cl - s\alpha sl$
$c\phi sl$	$s\phi s\alpha sl + c\alpha cl$	$-c\alpha c\phi sl + s\alpha cl$
$s\phi$	$-s\alpha c\phi$	$c\alpha\ c\phi$

where $c\phi$, $s\phi$ are the cosine and sine of latitude $c\alpha$, $s\alpha$ are the cosine and sine of alpha, and cl, sl are the cosine and sine of the change in longitude since TIME 0. Therefore

$$LAT = \tan^{-1}\left(\frac{COP\$2P(3,1)}{TEMP}\right) = \tan^{-1}\left(\frac{S\phi}{\sqrt{(c\phi)^2[(s\alpha)^2 + (c\alpha)^2]}}\right)$$

$$= \tan^{-1}\left(\frac{s\phi}{c\phi}\right)$$

where $TEMP = \sqrt{(COP\$2P\ (3,2))^2 + (COP\$2P(3,3))^2}$

i) if TEMP = 0, longitude and alpha cannot be extracted from the DCM so that LONG and ALPHA are not computed.

ii) if TEMP $\neq$ 0, longitude and alpha can be extracted as follows:

$$\text{LONG} = \tan^{-1}\left(\frac{\text{COP\$2P(2,1)}}{\text{COP\$2P (1,1)}}\right) + \text{LONG0}$$

$$= \tan^{-1}\left(\frac{c\phi s1}{c\phi c1}\right) + \text{LONG0}$$

$$= \tan^{-1}\left(\frac{s1}{c1}\right) + \text{LONG0}$$

where LONG0 is longitude at time = TIME0. Alpha is computed as follows:

$$\text{ALPHA} = \tan^{-1}\left(\frac{-\text{COP\$2P(3,2)}}{\text{COP\$2P(3,3)}}\right)$$

$$= \tan^{-1}\left(\frac{s\alpha c\phi}{c\alpha c\phi}\right) = \tan^{-1}\left(\frac{s\alpha}{c\alpha}\right)$$

C) Extrapolated values (to the middle of the next nav cycle) of the sine and cosine of alpha (XSALF and XCALF, respectively) are computed as follows:

DALPHA = (ALPHA -OALPHA) /2

where OALPHA is previous value of alpha.

XSALF = SALF * DALPHA * CALF

XCALF = CALF - DALPHA * SALF

(these are the first order approximations, eg. $\sin(\alpha + \dot{\alpha}t) = \sin(\alpha)\cos(\dot{\alpha}t) + \cos(\alpha)\sin(\dot{\alpha}t)$

$= \sin(\alpha) + \cos(\alpha)\,\dot{\alpha}t + 0\,((\dot{\alpha}t)^2)$

$\dot{\alpha}$ is being approximated as $\frac{\text{ALPHA-OALPHA}}{\text{DELT}}$,

t is $\frac{\text{DELT}}{2}$, this yields above approximation)

D) V$2 (the velocity in the 2 frame) is computed by rotating V$2P(2) and V$2P(3) through the alpha angle yielding V$2(2) and V$2(3).

Inputs, Outputs & Internal Variables.

Inputs:

LONG0 - Longitude at TIME0

C0P.$2P - Tranformation DCM

V$2P - Velocity in 2P frame

Outputs:

LAT - latitude

LONG - Longitude

ALPHA - alpha angle

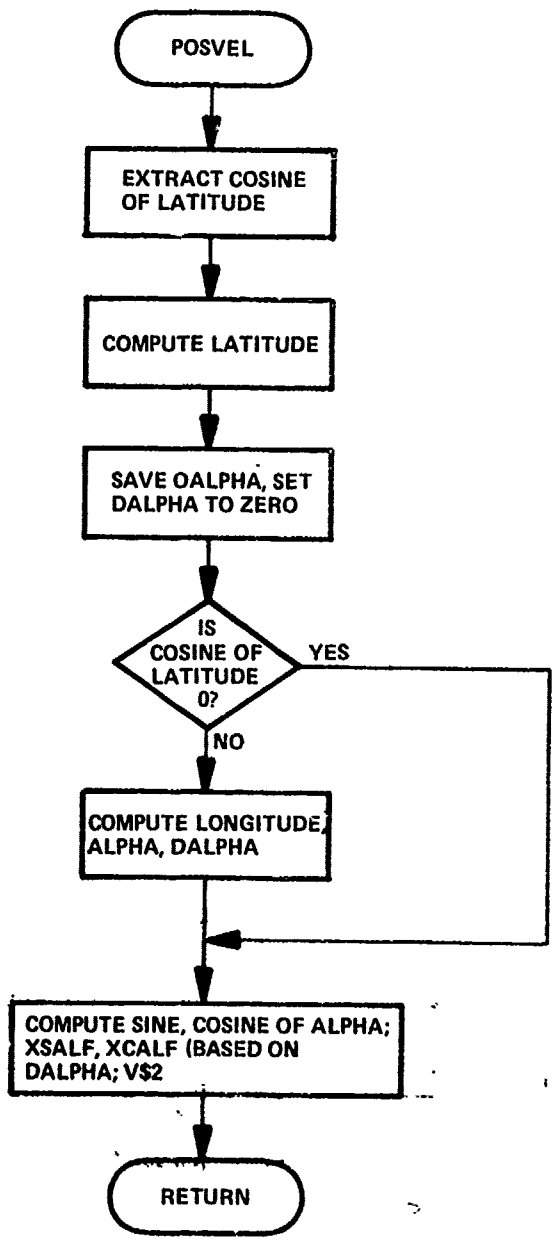
SALF - sine of ALPHA

CALF - cosine of ALPHA

XSALF - extrapolated sine of ALPHA

XCALF - extrapolated consine of ALPHA

V$2 - velocity in 2 frame


POSVEL
EXTRACT COSINE OF LATITUDE
COMPUTE LATITUDE
SAVE OALPHA, SET DALPHA TO ZERO
IS COSINE OF LATITUDE 0?
YES
NO
COMPUTE LONGITUDE, ALPHA, DALPHA
COMPUTE SINE, COSINE OF ALPHA; XSALF, XCALF (BASED ON DALPHA; V$2
RETURN

2.3.7 Function & Equations

The purpose of this subroutine is to compute the acceleration of gravity (as coordinatized in the 2P frame) and to compute the square of the sine of the latitude. These calculations are based on the inputs which are altitude (measured from the standard ellipsoid earth) and a DCM from 2P to 0P (which yields the position on the earth).

The sine squared of latitude is computed by picking up the sine of latitude out of the input DCM (row 3, column 1 is sine of latitude) and squaring it.

The vertical (geodetic) component of gravity is computed as

$$-(g + g_{\phi} \sin^2\phi + g_{\phi\phi} \sin^4 \phi) * (1 - (g_h - g_{h\phi}\sin^2\phi)\, h + g_{hh}\, h^2)$$

where g, g_{ϕ}, $g_{\phi\phi}$, g_h, g_{hh} and $g_{h\phi}$ are coefficients in the expansion of the analytic expression of gravity for an ellipsoid earth. The constants are based on the WGS72 survey.

The north component of gravity is computed as

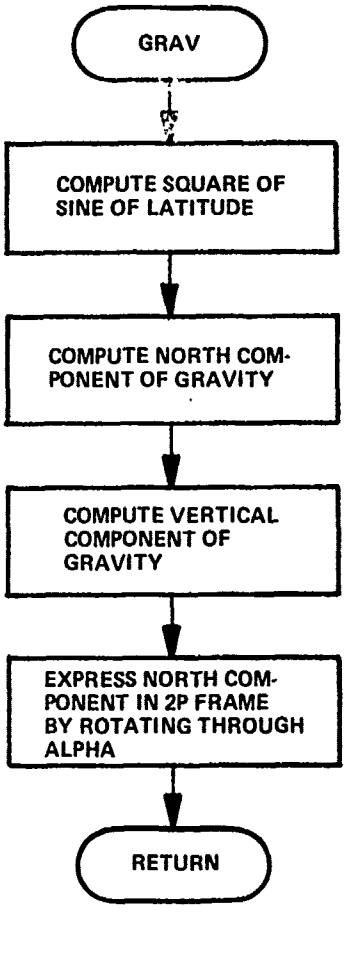
$-g_1 * h * \cos\phi * \sin\phi$ where h and ϕ are (as above) altitude and geodetic latitude, and g_1 is another WGS72 constant. The north component is then rotated through the alpha angle to be able to express it in the 2P frame. These calculations duplicate those of PROFGEN.

Inputs, Outputs of Internal Variables

A0P$2P - Input, DCM taking 2P to 0P
AH - Input, altitude
G$2P - Output, gravity vector
AS2PHI - Output, square of sine of latitude extracted from A0P$2P

GRCNT,
GRS2,
CRS4,
GRH,
GRHS3,
GRH2,
GLHSC
} Input, gravity coefficients

COEF - internal scratch variable.


GRAV
COMPUTE SQUARE OF SINE OF LATITUDE
COMPUTE NORTH COMPONENT OF GRAVITY
COMPUTE VERTICAL COMPONENT OF GRAVITY
EXPRESS NORTH COMPONENT IN 2P FRAME BY ROTATING THROUGH ALPHA
RETURN

ANGVL2

2.3.8 Function & Equations

The purpose of ANGVL2 is to compute RHO, the angular rate of the platform frame(2P) with respect to the earth-fixed frame (0P). These rates are computed based on an ellipsoid earth. The north and east velocities (VN and VE) are computed by taking the velocity in the 2P frame (as passed to the subroutine) and rotating it through the alpha angle (sine and cosine of alpha are passed to the subroutine).

The angular rates about east and north are computed as:

$$WE = \frac{-VN}{AH + RESQ * \left(\frac{1}{1 - ESQ * AS2PHI}\right)^{3/2}}$$

{ radius of curvature of earth in plane defined by north and up unit vectors }

$$WN = \frac{VE}{AH + RO * \left(\frac{1}{1 - ESQ * AS2PHI}\right)^{1/2}}$$

{ radius of curvature of earth in plane defined by east and up unit vectors }

Where AH is the input parameter representing altitude, AS2PHI is the input parameter of the square of the sine of latitude. RO is the equatorial earth radius, RESQ is RO * (1 - ESQ), where ESQ is the square of the earths' eccentricity, and where WE and WN are the rates (about east and north) needed to keep the up axis vertical. WE and WN are then rotated through the alpha angle, so that RHO is coordinatized in the 2P frame.

Inputs, Outputs & Internal Variables

VLCTY - input parameter representing velocities in 2P frame

ASALF - input parameter representing sine of alpha angle

ACALF - input parameter representing cosine of alpha angle

AS2PHI - input parameter representing square of sine of latitude

AH - input parameter representing altitude

VE - east component of VLCTY (internal variable)

VN - north component of VLCTY (internal variable)

WE - Rate, about east, of 2P frame with respect to OP (internal variable)

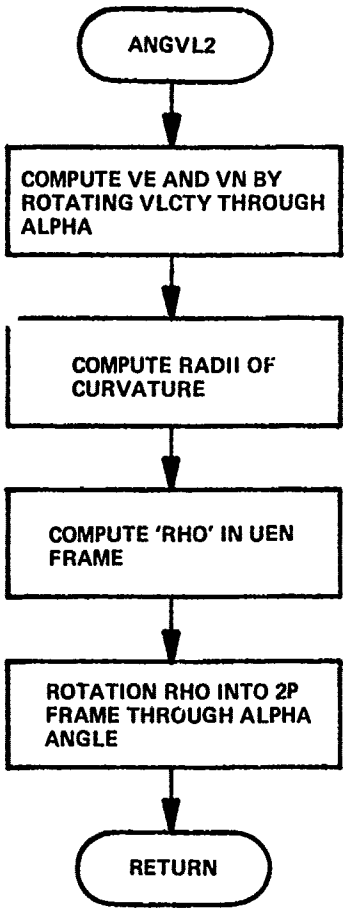
WN - Rate, about north, of 2P frame with respect to OP (internal variable)

RP - radius of curvature of earth in east direction (internal variable)

RM - radius of curvature of earth in north direction (internal variable)

RHO(2) - angular rate of 2P frame with respect to OP
RHO(3) frame, as seen in 2P frame (note that rate is about level axes only) - output variable

TMP1, - internal variables
TMP2


ANGVL2
COMPUTE VE AND VN BY ROTATING VLCTY THROUGH ALPHA
COMPUTE RADII OF CURVATURE
COMPUTE 'RHO' IN UEN FRAME
ROTATION RHO INTO 2P FRAME THROUGH ALPHA ANGLE
RETURN

2.3.9 Function & Equations

The purpose of ATUDE is to implement that part of the navigation algorithm which extracts the attitude angles (roll, pitch and heading) from C2P$5 and a knowledge of ALPHA.

Analytically, it is known that

$$C2P\$5 = \begin{bmatrix} cy * cz & -cy * sz & sy \\ cx*sz-sx*sy*cz & cx*cz+sx*sy*sz & sx * cy \\ -sx*sz-cx*sy*cz & -sx*cz+cx*sy*sz & cx * cy \end{bmatrix}$$

where
- sz = sine (roll)
- cz = cosine (roll)
- sy = sine (pitch)
- cy = cosine (pitch)
- sx = sine (yaw)
- cx = cosine (yaw)

$$\text{roll} = \tan^{-1}\left(\frac{-(-cy*sz)}{cy*cz}\right) = \tan^{-1}\left(\frac{-C2P\$5(1,2)}{C2P\$5(1,1)}\right)$$

$$\text{pitch} = \tan^{-1}\left(\frac{sy}{cy}\right) = \tan^{-1}\left(\frac{sy}{\sqrt{(sx*cy)^2 + (cx*cy)^2}}\right)$$

$$= \tan^{-1}\left(\frac{C2P\$5(1,3)}{\sqrt{(C2P\$5(2,3)^2 + (C2P\$5(3,3)^2}}\right)$$

$$\text{yaw} = \text{alpha} + \tan^{-1}\left(\frac{sx*cy}{cx*cy}\right) = \text{alpha} + \tan^{-1}\left(\frac{C2P\$5(2,3)}{C2P\$5(3,3)}\right)$$

Inputs, Outputs & Internal Variables

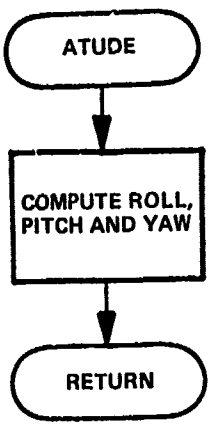
C2P$5 - DCM taking 5 frame to 2P frame (input)

ALPHA - alpha angle associated with 2 frame to 2P frame (input)

ETA(1) - roll (output)

ETA(2) - pitch (output)

ETA(3) - heading (output)



OUTUNI

2.3.10 Function & Equations

The purpose of OUTUNI is to perform all calculations needed to prepare data for output in man readable form. All angles are converted to degrees. The differences for output are found by subtracting simulator values from PROFGEN values. PROFGEN velocity is converted to (Up, East, North) local vertical north (LVN) frame for output as follows:

$$\begin{bmatrix} V\$2T(1) \\ V\$2T(2) \\ V\$2T(3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -SALF\$T & -CALF\$T & 0 \\ CALF\$T & -SALF\$T & 0 \end{bmatrix} \begin{bmatrix} V\$T(1) \\ V\$T(2) \\ V\$T(3) \end{bmatrix}$$

The variables are described below. The transformation matrix takes into account the change in axes plus the need to rotate α_T degrees in the opposite direction as in PROFGEN. IT is formed as shown below:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & CALF\$T & -SALF\$T \\ 0 & SALF\$T & CALF\$T \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Rotate $-\alpha_T$ degrees to put in LVN frame

change of axes fror. NWU to UEN

Input Variables

RADPER - Factor for converting from radians to degrees.

LAT$T - Geodetic latitude from PROFGEN.

LONG$T - Geodetic longitude from PROFGEN.

ALF$T - Wander angle ($\alpha_T$) from PROFGEN.

LAT - Geodetic latitude from simulator.

LONG - Geodetic longitude from simulator.

ALPHA - Wander angle (α_T) from simulator.

ETA$T - Attitude angles from PROFGEN.

ETA - Attitude angles from simulator.

V$T - Aircraft velocity (NWU) in LLWA from PROFGEN.

V$2 - Aircraft velocity (UEN) in LVN from simulator.

HB - Aircraft altitude from PROFGEN.

H - Aircraft altitude from simulator.

SALF$T- Sine of AFL$T.

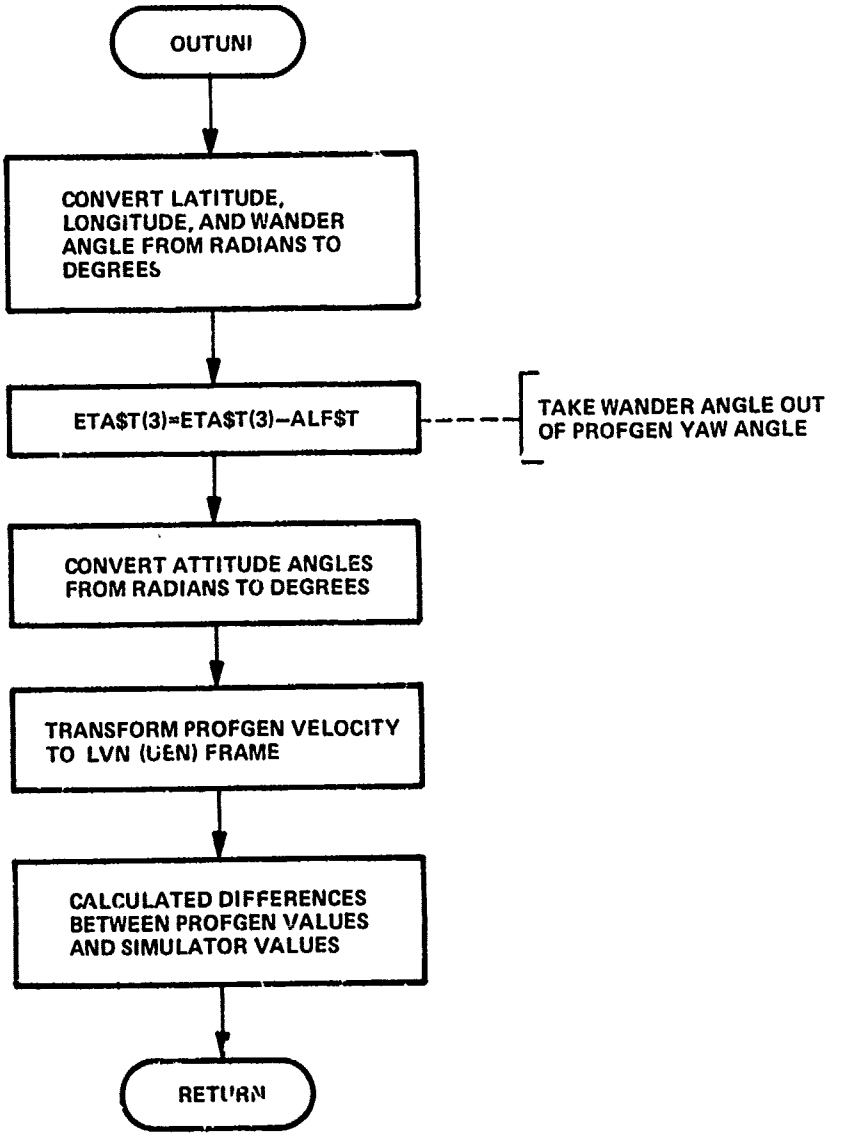
CALF$T- Cosine of ALF$T.

Internal Variables

None

Output Variables

OLAT$T - LAT$T in degrees.
OLON$T - LONG$T in degrees.
OALF$T - ALF$T in degrees.
OLAT - LAT in degrees.
OLONG - LONG in degrees.
OALF - ALPHA in degrees.
OETA$T - ETA$T in degrees.
OETA - ETA in degrees.
V$2T - Aircraft velocity from PROFGEN transformed to LVU (UEN) frame.
DH - Difference in altitude.
DLAT - Difference in geodetic latitude.
DLONG - Difference in geodetic longitude.
DALF - Difference in wander angle.
DV - Difference in aircraft velocity in LVN frame.
DETA - Difference in aircraft atitude.


OUTUNI
CONVERT LATITUDE, LONGITUDE, AND WANDER ANGLE FROM RADIANS TO DEGREES
ETA$T(3)=ETA$T(3)–ALF$T
TAKE WANDER ANGLE OUT OF PROFGEN YAW ANGLE
CONVERT ATTITUDE ANGLES FROM RADIANS TO DEGREES
TRANSFORM PROFGEN VELOCITY TO LVN (UEN) FRAME
CALCULATED DIFFERENCES BETWEEN PROFGEN VALUES AND SIMULATOR VALUES
RETURN

PRINTR

2.3.11 Function & Equations

Thw purpose of PRINTR is to output the standard output variables on a periodic basis. The standard variables are:

Time
Latitude (geodetic)
Longitude (geodetic)
Alpha (wander angle)
Altitude
Velocity (local vertical north)
Attitude

The operator specifies the period between printout.

Input Variables

INIT - Signals first pass through simulator. "0" on first pass.
IPRINT - Number of lines already printed on page.
IPLIM - Limit on number of printed lines per page.
ITITLE - 80 character operator specified title.
IPAGE - Page number.
TIME - Simulation time (elapsed time).
IPC(25)- Operator controlled flag used to control output of filtered attitude data. "0" value gives no printout. Default value is "0".
FILATT - Filtered attitude data 3x3 array containing the angle, angular rate, and angular acceleration for roll, pitch, and yaw.

The following list is grouped by threes, the first variable in each group is from PROFGEN. The second variable is from the simulator, and the third is the difference between the two.

OLAT$T, OLAT, DLAT - Geodetic latitude.
OLON$T, OLONG, DLONG - Geodetic longitude.
OALF$T, OALF, DALF - Wander angle.
HB, H, DH - Altitude
V$2T, V$2, DV - Velocity in local vertical north.
OETA$T, OETA, DETA - Attitude angles.

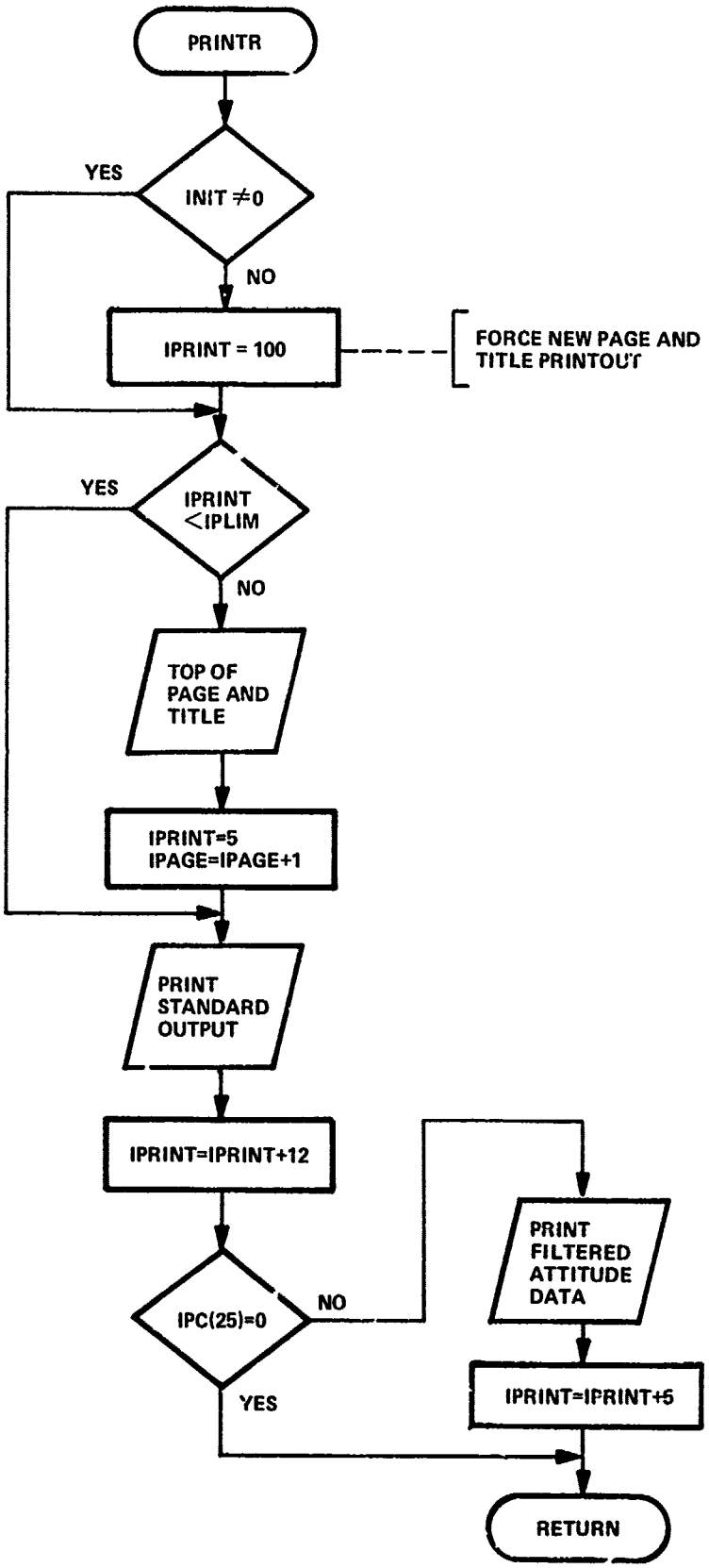
Internal Variables

None

Output Variables

IPAGE - Printout page number.

IPRINT - Number of lines printed on page.


PRINTR
YES
INIT ≠0
NO
IPRINT = 100
FORCE NEW PAGE AND TITLE PRINTOUT
YES
IPRINT <IPLIM
NO
TOP OF PAGE AND TITLE
IPRINT=5
IPAGE=IPAGE+1
PRINT STANDARD OUTPUT
IPRINT=IPRINT+12
IPC(25)=0
NO
YES
PRINT FILTERED ATTITUDE DATA
IPRINT=IPRINT+5
RETURN

2.3.12 Function & Equations

The purpose of PLTAPE is to create a tape suitable for use as input to plotting software. The data is output in fixed length records 22 words per record, containing the variables:

Time
Latitude
Longitude
Wander angle
Altitude
Velocity (LVN frame)
Attitude (roll, pitch, yaw)

All the above items represent the difference between simulator values and PROFGEN values. The operator can specify the output period via card input at execution time. A copy of the operator specified title is output as the first record on the tape.

Input Variables

INIT - Flag which indicates first pass (when equal 0) through simulator.
ITITLE - Operator specified title: 80 characters.
TIME - Elapse time

The following variables give the difference between the simulator generated value and the value from PROFGEN.

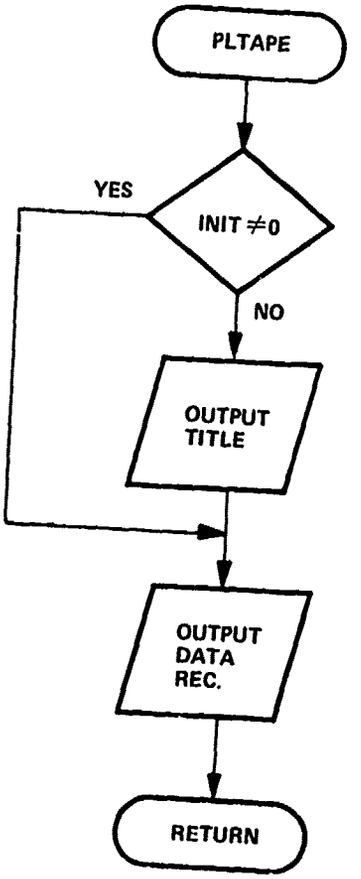
DLAT - Geodetic latitutde.
DLONG - Geodetic longitude.
DALF - Wander angle.
DH - Altitude.
DV - Velocity in local vertical north. Frame (U,E,N).
DETA - Attitude angles (roll, pitch, yaw).

Internal Variables

None

Output Variables

None



2.3.13 Function and Equations

The purpose of DOUT is to give the user the capability of printing variables other than those from the standard output set. This can be used for diagnostic or comparison purpose. Output is selected by a number which is passed in the calling sequence as shown below:

CALL DOUT (J)

The variable J is used in a computed GO TO in order to select the desired output. The output variables and the J value associated with each are described below. J values 20 through 30 are unused by DOUT.

Input Variables

- IPRINT - Number of lines already printed on page
- IPAGE - Next page number.
- WT - Angular rates about LLWA axes (J=1).
- ETA$P - Four gimbal angles for local-level and space stable. Incremental angles from gyros for strap-down (J=2).
- DV$2P - Delta velocity for use as input to navigation equations (J=3).
- V$2P - Instantaneous ground velocity as computed in the navigation equations (J=4).
- RHO - Angular rate of craft with respect to earth fixed frame; about level axes only (J=5).
- COP$2P - Direction cosine matrix for transformations from LLWA frame (2P) to earth fixed frame (0P) (J=6 or 7).
- THTERT - Rotation of earth with respect to inertial space during one computation cycle (J=8).
- THTTRQ - Commanded torque; rotation need to keep LLWA frame (2P) level (J=9).
- WXV - $\underline{\omega} \times \underline{v}$ terms of velocity update equation (J=10).
- VDMP - Vertical velocity damping term (J=11).
- G$2P - Gravity vector in LLWA frame. (J=12).

A(1,2)
A(2,3)
A(3,1) } Incremental misalignment angle used to update misalignment matrix (J=13).

C10$2P - Transformation matrix from LLWA frame (2P) to space stable frame (10) (J=14).

C2P$5 - Transformation matrix from body frame (5) to LLWA frame (2P) (J=15).

CO$5 - Transformation matrix from body frame (5) to inertial frame (0) (J=16).

TIME - Elapsed time in seconds (J=17).

Q2P$5 - Quaternion representation of rotation in transformation from body frame (5) to LLWA frame (2P) (J=19).

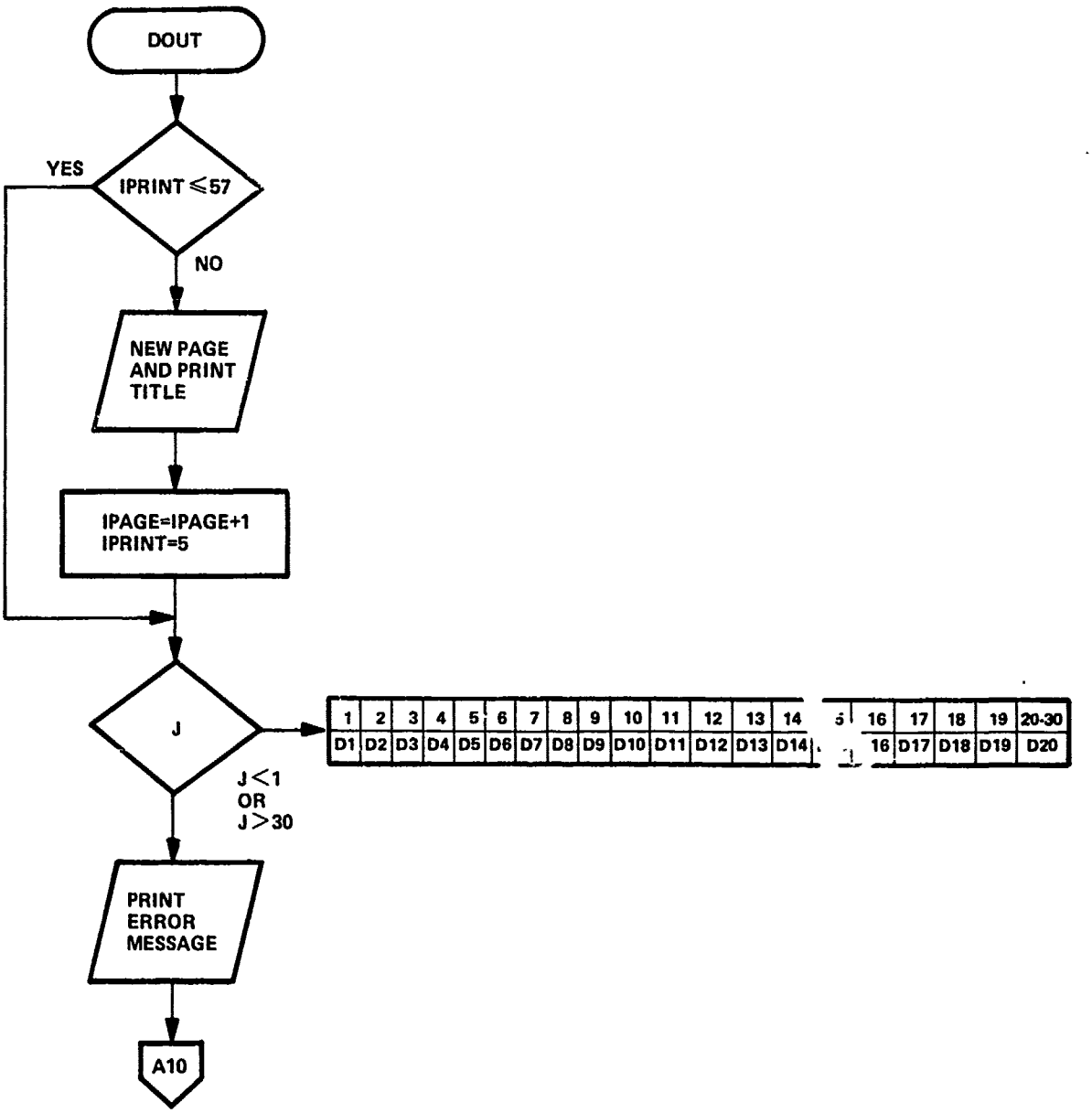
Internal Variables

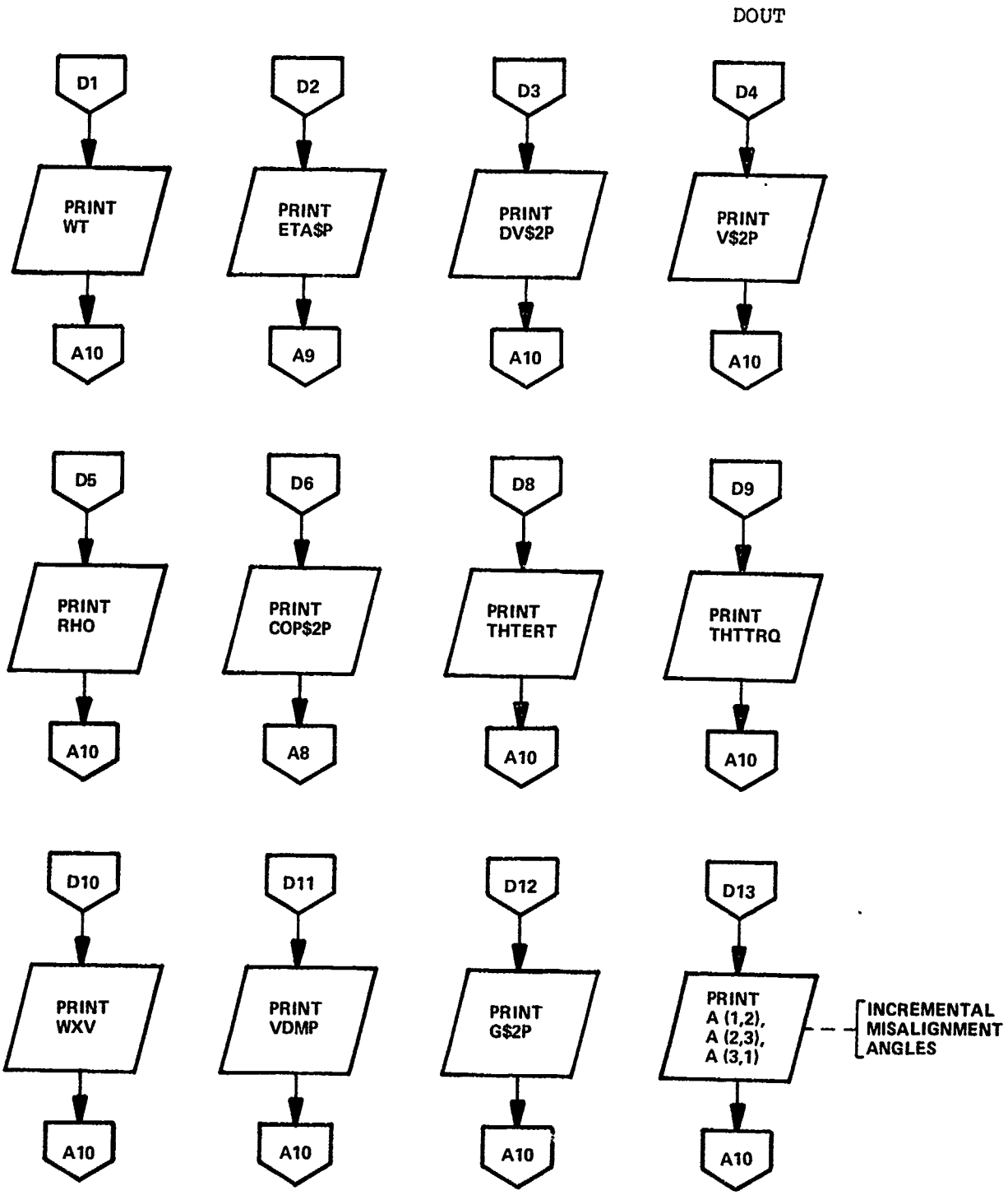
TMPARR - Elements (3,2), -(3,1), and (2,1) of $(C2P\$5)^{T} \cdot (C2P\$5T)$. Printed when J=18.

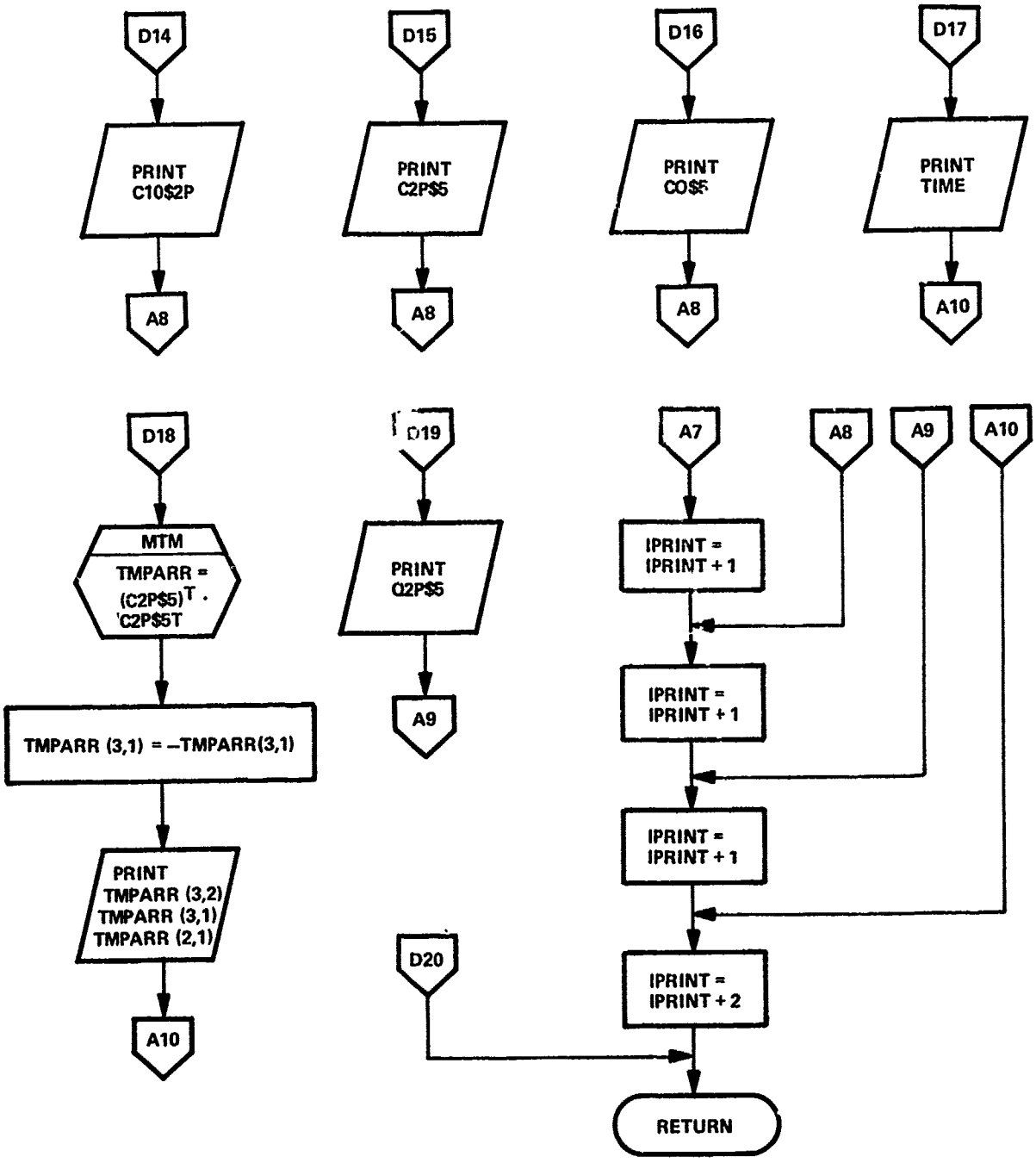
Output Variables

IPRINT - Number of lines already printed on page.

IPAGE - Page number of next page.


DOUT
YES
IPRINT ≤57
NO
NEW PAGE AND PRINT TITLE
IPAGE=IPAGE+1
IPRINT=5
J
J<1 OR J>30
PRINT ERROR MESSAGE
A10
1 2 3 4 5 6 7 8 9 10 11 12 13 14
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14
16 17 18 19 20-30
16 D17 D18 D19 D20


DOUT
D1
PRINT WT
A10
D2
PRINT ETA$P
A9
D3
PRINT DV$2P
A10
D4
PRINT V$2P
A10
D5
PRINT RHO
A10
D6
PRINT COP$2P
A8
D8
PRINT THTERT
A10
D9
PRINT THTTRQ
A10
D10
PRINT WXV
A10
D11
PRINT VDMP
A10
D12
PRINT G$2P
A10
D13
PRINT A (1,2), A (2,3), A (3,1)
INCREMENTAL MISALIGNMENT ANGLES
A10


D14
PRINT
C10$2P
A8
D15
PRINT
C2P$5
A8
D16
PRINT
CO$5
A8
D17
PRINT
TIME
A10
D18
MTM
TMPARR =
(C2P$5)T .
'C2P$5T
TMPARR (3,1) = –TMPARR(3,1)
PRINT
TMPARR (3,2)
TMPARR (3,1)
TMPARR (2,1)
A10
D19
PRINT
C2P$5
A9
A7
A8
A9
A10
IPRINT =
IPRINT + 1
IPRINT =
IPRINT + 1
IPRINT =
IPRINT + 1
IPRINT =
IPRINT + 2
D20
RETURN

ININIT

2.3.14 Function & Equations

The purpose of ININIT is to initialize variables used by INRNAV (which implements the navigation equations). If "higher order algorithm" (IPC(29)≠0) is selected those variables needed by the "higher order algorithm" in INRNAV are also initalized. The equations are the same as those used by INRNAV.

Inputs, Outputs and Internal Variables

Variables initialized are as follows (* implies initialization only when IPC(29)≠0, ⊕ implies recomputed by POSVEL)

	LONG0	All initialized to TIME0 PROFGEN values
	TIME0	
⊕	LAT	
⊕	LONG	
⊕	ALPHA	
⊕	CALF	
⊕	SALF	
	C0P$2P	
	H	
	OHB	
	HB	
	S2PHI	
	V$2P	
*	HV$2P	
*	XH	
*	OV$2P	
*	XS2PHI	
⊕*	XSALF	All initialized to TIME0 PROFGEN values
⊕*	XCALF	
*	XV$2P	

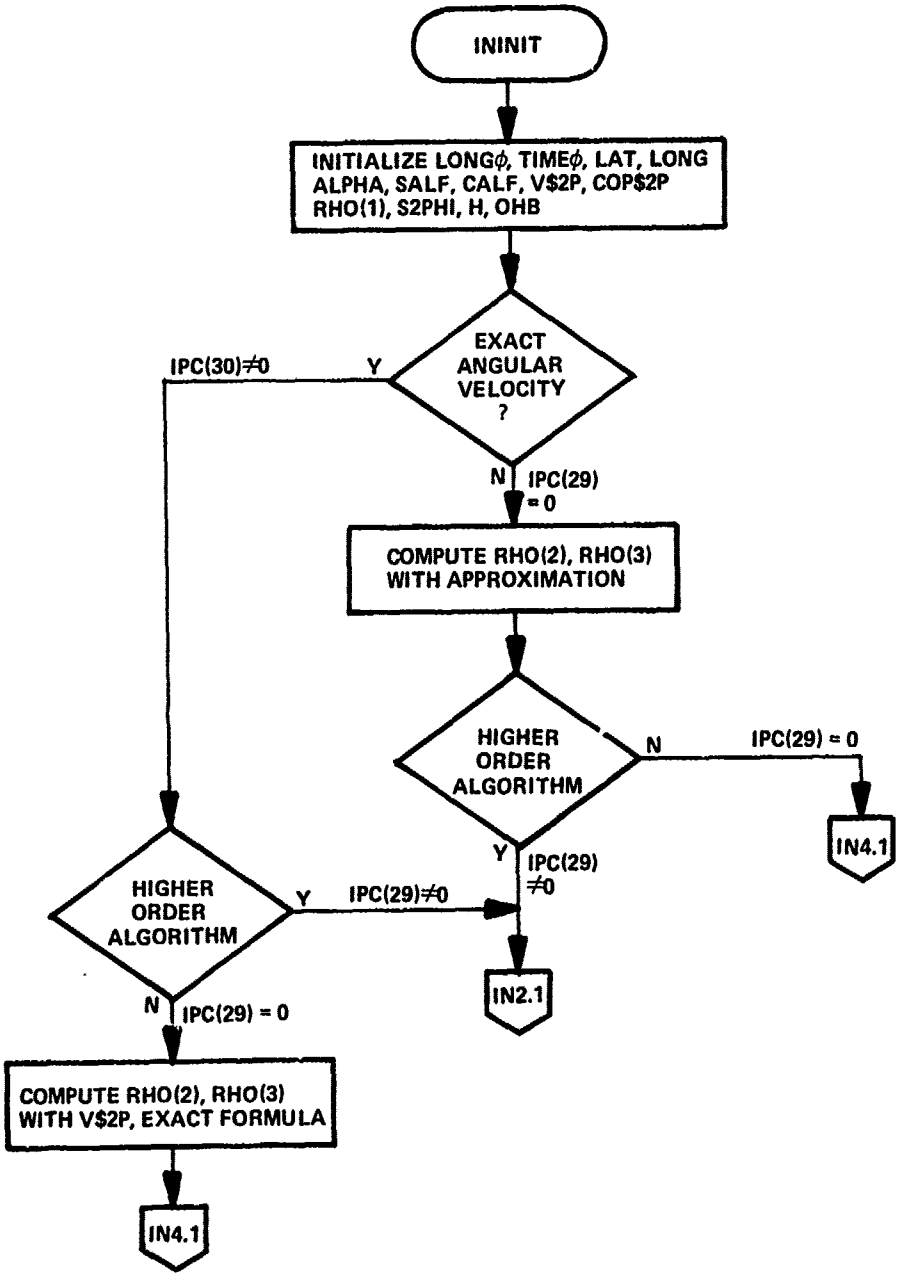
RHO
C$RHO
C$2P
WXV
THTTRQ
THTCOR
THTERT
DALPHA
V$2

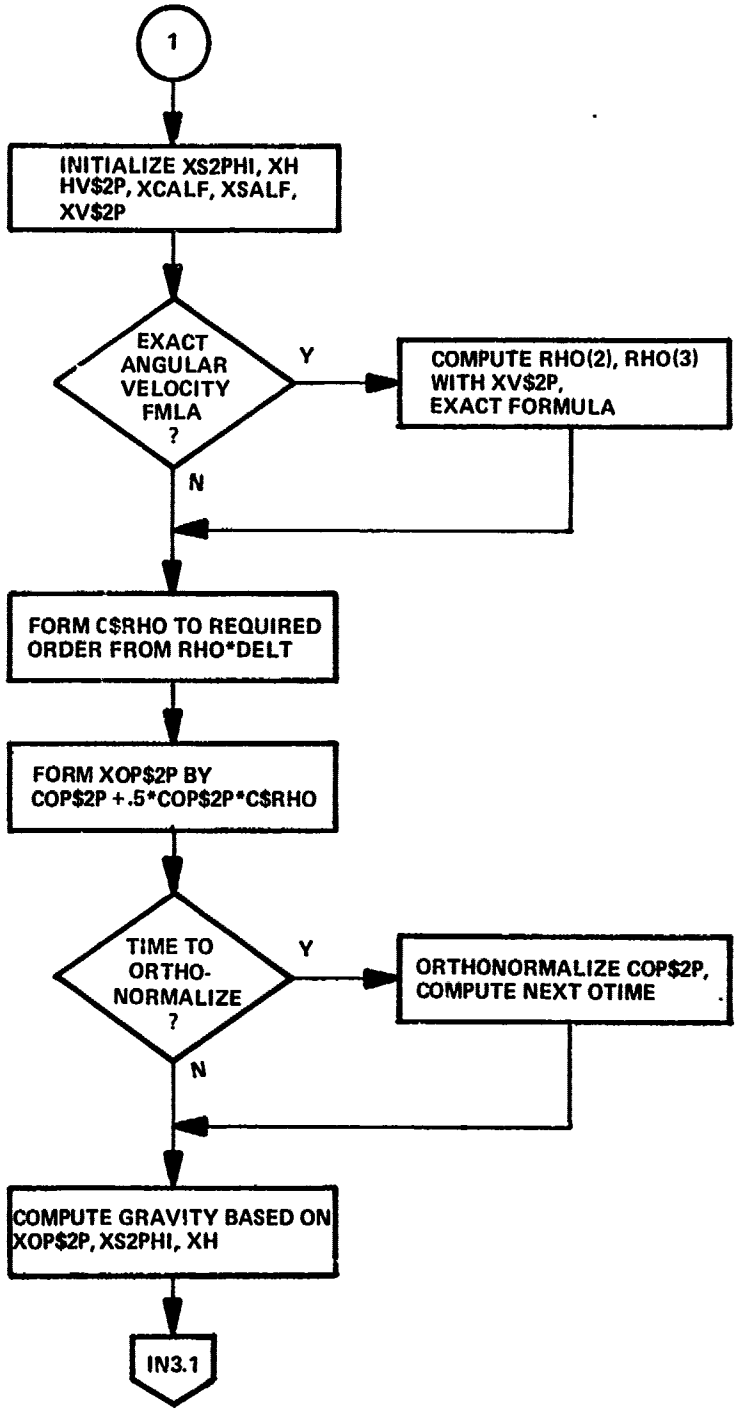
Initialized according to values of IPC(30), IPC(29) and IPC (28), identically as INRNAV would compute

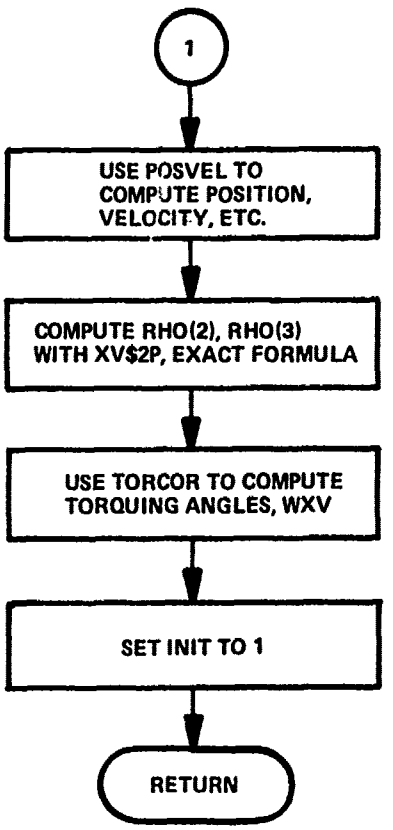
INIT set to 1 to indicate end of initialization
OTIME is recomputed if orthonormalization takes place during ININIT
Internal variables (all scratch): I, J, DCM32S, DEM33S CORR, ONEM

Inputs:

A is used as scratch
Universal constants (common block UNVRSL) are used.
INCOR
DELT


ININIT
INITIALIZE LONGϕ, TIMEϕ, LAT, LONG
ALPHA, SALF, CALF, V$2P, COP$2P
RHO(1), S2PHI, H, OHB
EXACT ANGULAR VELOCITY ?
IPC(30)≠0
Y
N
IPC(29) = 0
COMPUTE RHO(2), RHO(3) WITH APPROXIMATION
HIGHER ORDER ALGORITHM
N
IPC(29) = 0
IN4.1
Y
IPC(29) ≠0
HIGHER ORDER ALGORITHM
Y
IPC(29)≠0
IN2.1
N
IPC(29) = 0
COMPUTE RHO(2), RHO(3) WITH V$2P, EXACT FORMULA
IN4.1


1
INITIALIZE XS2PHI, XH HV$2P, XCALF, XSALF, XV$2P
EXACT ANGULAR VELOCITY FMLA ?
Y
COMPUTE RHO(2), RHO(3) WITH XV$2P, EXACT FORMULA
N
FORM C$RHO TO REQUIRED ORDER FROM RHO*DELT
FORM XOP$2P BY COP$2P + .5*COP$2P*C$RHO
TIME TO ORTHO-NORMALIZE ?
Y
ORTHONORMALIZE COP$2P, COMPUTE NEXT OTIME
N
COMPUTE GRAVITY BASED ON XOP$2P, XS2PHI, XH
IN3.1


1
USE POSVEL TO COMPUTE POSITION, VELOCITY, ETC.
COMPUTE RHO(2), RHO(3) WITH XV$2P, EXACT FORMULA
USE TORCOR TO COMPUTE TORQUING ANGLES, WXV
SET INIT TO 1
RETURN

CMINTG

2.3.15 Function & Equations

A. The purpose of CMINTG is to form up the ΔV's that would be sensed by the accelerometers (in a SS, LL or SD IMU) over the time period between samples of simulated accelerometer data. The routine also quantizes (makes allowance for the finite resolution of) the accelerometer data. In addition the routine checks for discontinuities generated by PROFGEN and takes special action when detected. The input to the routine is in the form of instantaneous sensed accelerations (specific forces).

B. We want $\Delta V_x = \int_t^{t+n\Delta t} a_x(t)dt$. (Similarly for ΔV_y and ΔV_z). n is passed to CMINTG as ICMCYL (CMINTG cycle length). $a_x(t)$ are the instantaneous accelerations. Δt is DELTS.

i) For ICMCYL odd ΔV_x is computed using the trapezoidal rule, that is

$$\sum_{i=1}^{n}\left[\left(a_x(t + (i-1)\Delta t) + (t + i\Delta t)\right)/2.\right]$$

ii) For ICMCYL even ΔV_x is computed using Simpson's rule, that is

$$n\Delta t[a_x(t) + 2a_x(t+\Delta t) + 4a_x(t+2\Delta t) + 2a_x(t+3\Delta t) + \ldots\ldots$$
$$4a_x(t + (n-2)\Delta t) + 2a_x(t+(n-1)\Delta t) + a_x(t+n\Delta t)]/(3n-2)$$

C. The above yields the total ΔV. We must now quantize (truncate to allow for finite resolution of the accelerometers)

$\underline{TEMP} = \lfloor \underline{\Delta V} + \underline{R^*} \rfloor$

($\underline{R^*}$ are residuals from previous quantization) $\lfloor \underline{x} \rfloor$ means round $\underline{x}$ down to an integral number of units)

$\underline{R} = \underline{\Delta V} + \underline{R^*} - \underline{TEMP}$

$\underline{\Delta V} = \underline{TEMP}$

D. PROFGEN, under certain circumstances, generates discontinuities in its acceleration. CMINTG performs the following check for the x-component (similarly for y and z)

$$(a_x(t) - a_x(t-\Delta t))/\Delta t > TOLJRK$$

If this is true (on any axis) $a_x(t)$ is set equal to $a_x(t-\Delta t)$ and a warning message is printed.

Inputs, Outputs and Internal Variables

A. Inputs

SF$TP - previous value (that is, from TIME-DELTS) of acceleration in accelerometer frame

SF$T - current value of instantaneous acceleration in accelerometer frame

TOLJRK - tolerance for 'jerk'

ICMCYL - Number of DELTS intervals over which CMINTG is integrating

TIME - simulated time

ICMCYP - Number (that is 1 thru ICMCYL) of the interval that CMINTG is currently processing

DV$P - Integrated value of SF$TS

RESID - Residual value from previous cycle

VQUANT - Velocity quantization units in ft/sec

DELTS - Sample interval

B. Outputs

SF$TP - next 'previous value'

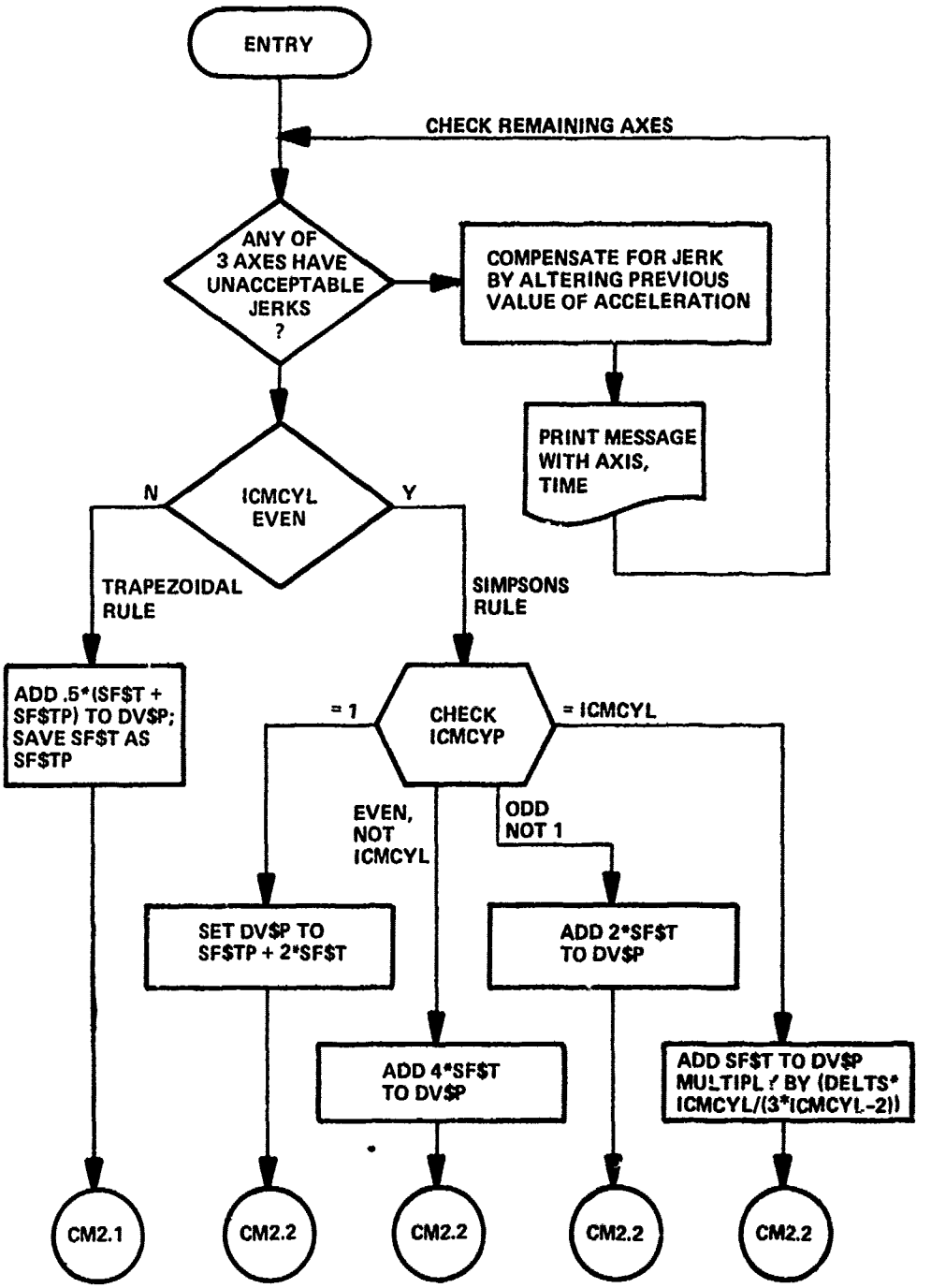
ICMCYP - next 'number of interval'

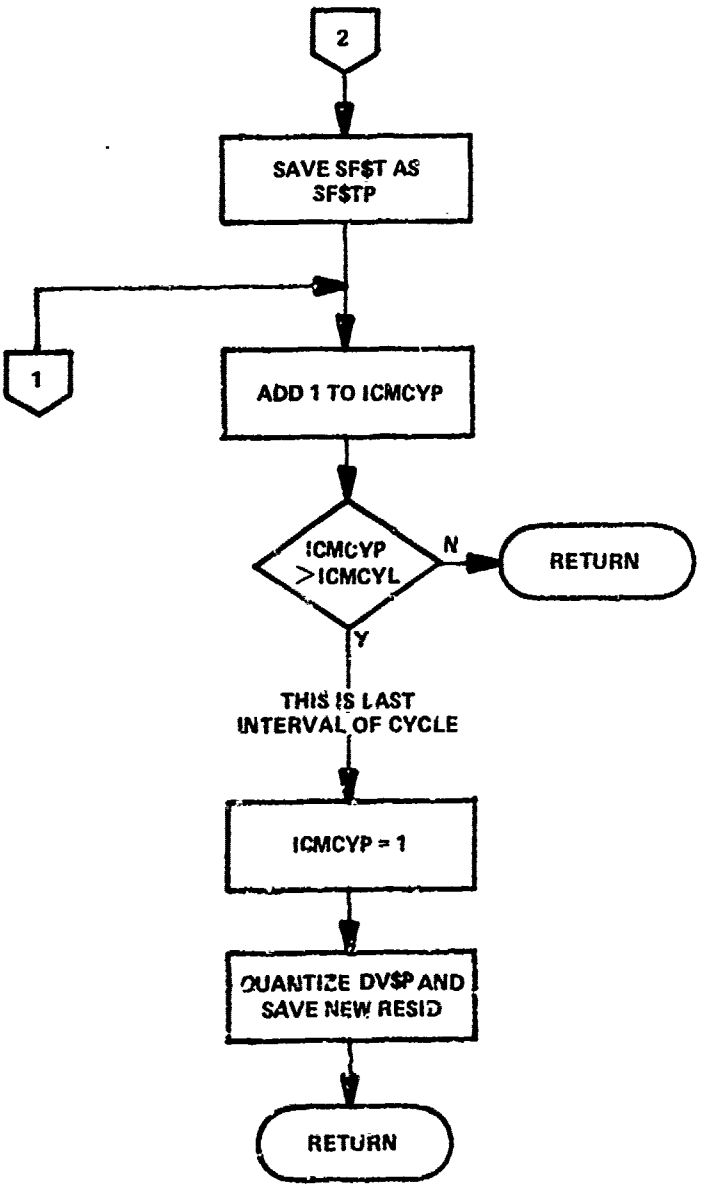
DV$P - updated 'integrated value of SF$T'

RESID - Residual for next cycle

C. Interval Variables

I - index scratch


ENTRY
CHECK REMAINING AXES
ANY OF 3 AXES HAVE UNACCEPTABLE JERKS ?
COMPENSATE FOR JERK BY ALTERING PREVIOUS VALUE OF ACCELERATION
PRINT MESSAGE WITH AXIS, TIME
N
ICMCYL EVEN
Y
TRAPEZOIDAL RULE
SIMPSONS RULE
ADD .5*(SF$T + SF$TP) TO DV$P; SAVE SF$T AS SF$TP
= 1
CHECK ICMCYP
= ICMCYL
EVEN, NOT ICMCYL
ODD NOT 1
SET DV$P TO SF$TP + 2*SF$T
ADD 2*SF$T TO DV$P
ADD 4*SF$T TO DV$P
ADD SF$T TO DV$P MULTIPL ? BY (DELTS* ICMCYL/(3*ICMCYL-2))
CM2.1
CM2.2
CM2.2
CM2.2
CM2.2


2
SAVE SF$T AS SF$TP
1
ADD 1 TO ICMCYP
ICMCYP >ICMCYL
N
RETURN
Y
THIS IS LAST INTERVAL OF CYCLE
ICMCYP = 1
QUANTIZE DV$P AND SAVE NEW RESID
RETURN

2.3.16 Function & Equations

The purpose of ATTFIL is to perform the calculations necessary to implement a filter for the attitude data. The filter operates by keeping estimated values of nine quantities, the roll, pitch, and yaw, and the first and second derivatives (with respect to time). These nine values are known as the "state matrix". The steps performed are as follows:

1. Extrapolate the state matrix to the current time (the state matrix, on entry to the attitude filter, has the values of the state matrix at the end of the previous filter cycle) as follows:

$$\text{New} = \text{Transition} * \text{Old}$$

$$\begin{bmatrix} \theta & \phi & \psi \\ \dot{\theta} & \dot{\phi} & \dot{\psi} \\ \ddot{\theta} & \ddot{\phi} & \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta & \phi & \psi \\ \dot{\theta} & \dot{\phi} & \dot{\psi} \\ \ddot{\theta} & \ddot{\phi} & \ddot{\psi} \end{bmatrix}$$

where θ, ϕ, ψ and Δt are roll, pitch, yaw and attitude cycle.

2. Compute residuals, that is, difference between measured θ, ϕ, and ψ and extrapolated values.

$$\begin{bmatrix} \theta_r \\ \phi_r \\ \psi_r \end{bmatrix} = \begin{bmatrix} \theta_m \\ \phi_m \\ \psi_m \end{bmatrix} - \begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix}$$

where θ_r and θ_m are residual roll and measured roll; likewise ϕ_r, ϕ_m, ψ_r, ψ_m.

3. For each of roll, pitch, and yaw select one of ten columns out of a 3-by-10 matrix of "filter gains". These columns are ordered so that moving towards the right (that is, increasing the column number) yields columns that have

increasing reliance on the "history" as reflected in the values of the first and second derivatives. The details of the selection process are as follows:

a) Compares the residual against a limit (stored as RSMAX), if greater "move left" by decreasing the column index (One such index is associated with each of roll, pitch and yaw). Otherwise

b) Compares the magnitude of the difference between the old residual and the new residual against a limit (stored as DRSMAX), if greater "move left" by decreasing the column index, otherwise

c) "Move right" by increasing the column index.

If an index is at column 10 and wishes to "move right", one stays at column 10. If an index is at column 1 and wishes to "move left", one stays at column 1, but must clear the corresponding first and second derivatives.

4. Update the "state matrix" with the residuals by using the selected gains.

$$\begin{bmatrix} \theta & \phi & \psi \\ \dot{\theta} & \dot{\phi} & \dot{\psi} \\ \ddot{\theta} & \ddot{\phi} & \ddot{\psi} \end{bmatrix} = \begin{bmatrix} K_{1\theta} & K_{1\phi} & K_{1\psi} \\ K_{2\theta} & K_{2\phi} & K_{2\psi} \\ K_{3\theta} & K_{3\phi} & K_{3\psi} \end{bmatrix} \begin{bmatrix} \theta_r & 0 & 0 \\ 0 & \phi_r & 0 \\ 0 & 0 & \psi_r \end{bmatrix} + \begin{bmatrix} \theta & \phi & \psi \\ \dot{\theta} & \dot{\phi} & \dot{\psi} \\ \ddot{\theta} & \ddot{\phi} & \ddot{\psi} \end{bmatrix}$$

where $K_{1\theta}$ $K_{2\theta}$ $K_{3\theta}$ are gains from column selected for θ; likewise for $K_{1\phi}$, $K_{2\phi}$, $K_{3\phi}$, $K_{1\psi}$, $K_{2\psi}$, $K_{3\psi}$.

Inputs, Outputs and Internal Variables

Inputs:

ICP(25) - 0 implies filter has not been requested. 1 implies filter has been activated.

IGAINP - 3 element array containing column indices for each of roll, pitch and yaw respectively

RSMAX, DRSMAX - tolerances

OFILTR - Old filter residuals, that is filter residuals from previous cycle

ETA - Attitude computed by simulator the attitude "measurement" for the filter.

FILATT - contains state matrix at end of previous cycle

Outputs:

OFILTR

IGAINP as above

FILATT

Interval Variables

A - holds extrapolated state matrix

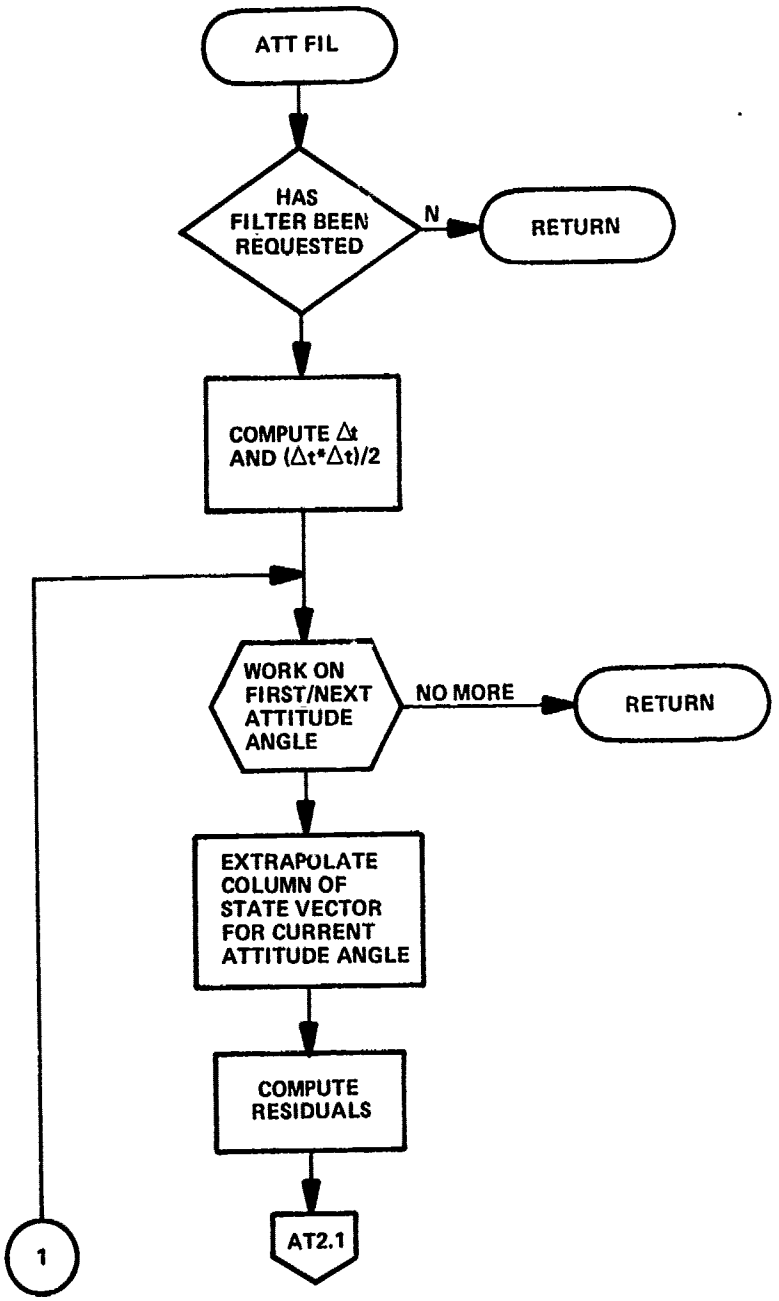
DT - Δt above

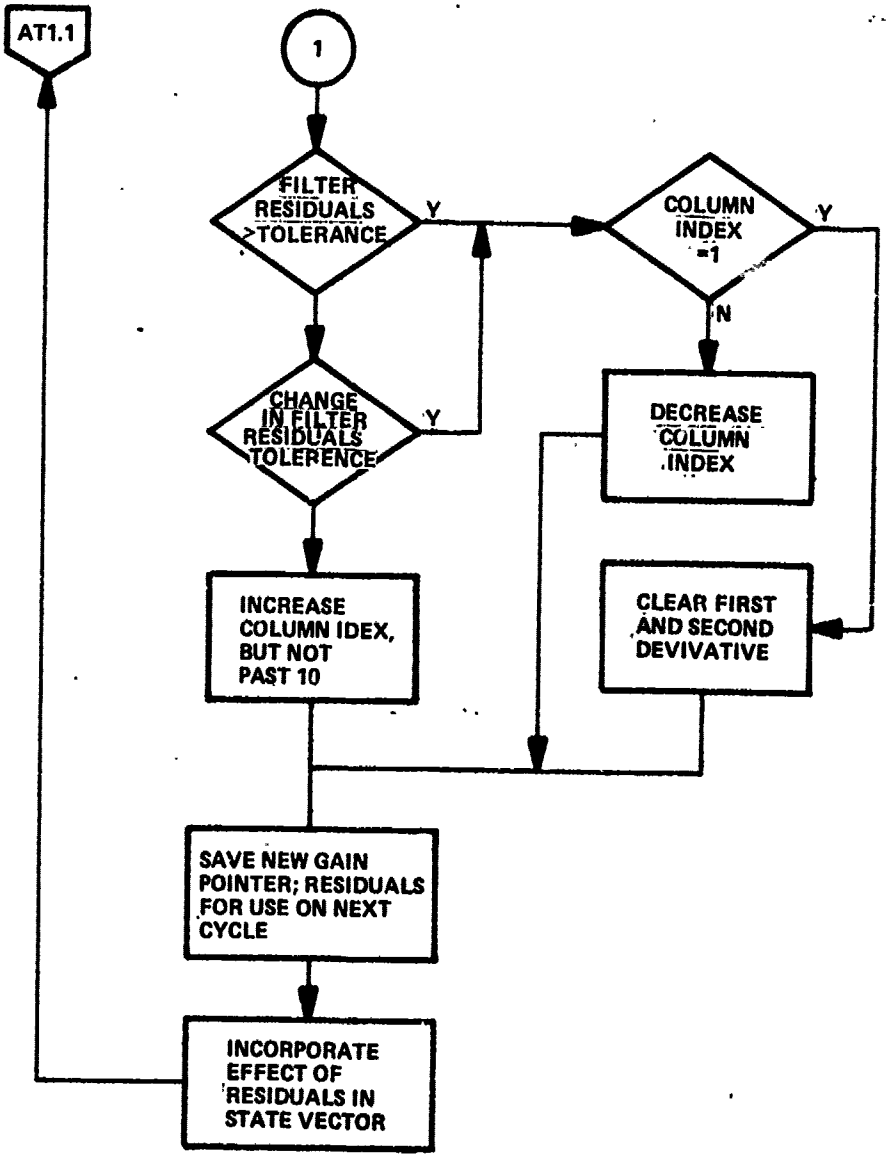
FILTR - holds filter residuals until stored in OFILTR

DTSQ2 - $\Delta t * \Delta t/2$

I,J,K - index scratch

CHECK IPC(25)


ATT FIL
HAS FILTER BEEN REQUESTED
N
RETURN
COMPUTE Δt AND (Δt*Δt)/2
WORK ON FIRST/NEXT ATTITUDE ANGLE
NO MORE
RETURN
EXTRAPOLATE COLUMN OF STATE VECTOR FOR CURRENT ATTITUDE ANGLE
COMPUTE RESIDUALS
AT2.1
1


AT1.1
1
FILTER RESIDUALS >TOLERANCE
Y
COLUMN INDEX =1
Y
N
CHANGE IN FILTER RESIDUALS TOLERENCE
Y
DECREASE COLUMN INDEX
INCREASE COLUMN IDEX, BUT NOT PAST 10
CLEAR FIRST AND SECOND DEVIVATIVE
SAVE NEW GAIN POINTER; RESIDUALS FOR USE ON NEXT CYCLE
INCORPORATE EFFECT OF RESIDUALS IN STATE VECTOR

SDINIT

2.3.17 Function & Equations

The purpose of SDINIT is to perform initialization when strapdown IMU is selected (IMUTYP = 3). The sequence of initializations performed is as follows (note that much initialization is performed in the BLOCK DATA routine):

a) INREC is called . This initializized all variables in the TRAJIN common block, in addition to all rates (DELT, DELTS, etc.): ICMCYL is initialized so that integration cycle is the attitude cycle.

b) COP$2P is initialized.

c) C2P$5 is initialized.

d) CO$5 is initialized.

e) SDINTG is called to transform the initial acceleration vector (SF$T) into the platform frame; this vector is then stored as SF$TP.

f) An initialization message is printed.

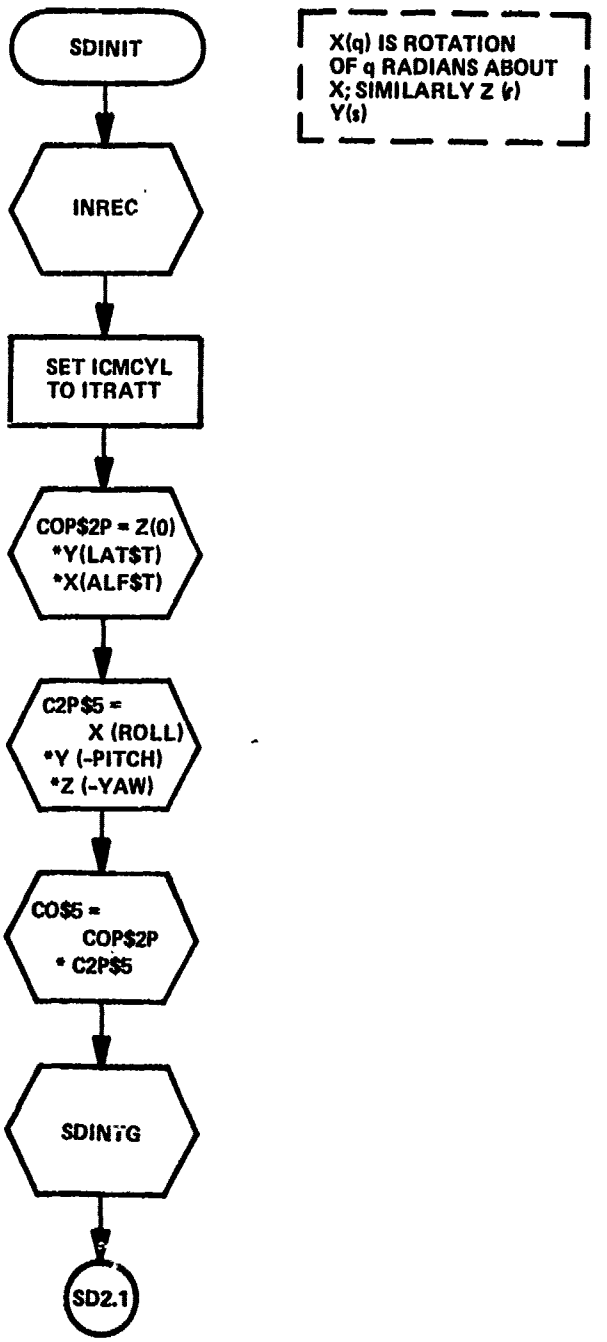
g) ININIT is called to initialize the navigation equations (see ININIT documentation).

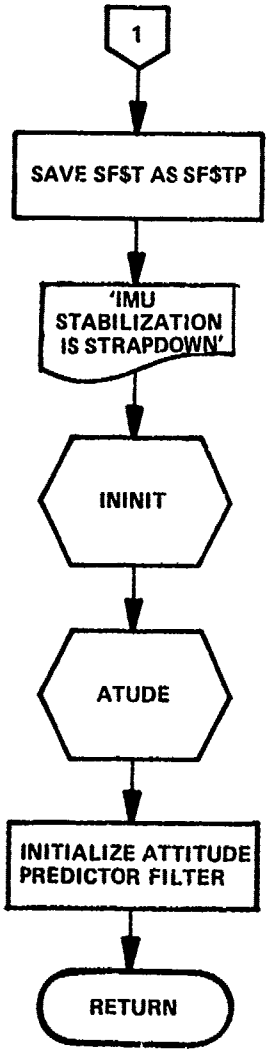
h) ATUDE is called to get initial ETAs.

i) Attitude predictor-filter is initialized.

Inputs, Output & Internal Variables

All variables not previously initialized are initialized.


SDINIT
INREC
SET ICMCYL TO ITRATT
COP$2P = Z(0) *Y(LAT$T) *X(ALF$T)
C2P$5 = X (ROLL) *Y (-PITCH) *Z (-YAW)
CO$5 = COP$2P * C2P$5
SDINTG
SD2.1
X(q) IS ROTATION OF q RADIANS ABOUT X; SIMILARLY Z (r) Y(s)


1
SAVE SF$T AS SF$TP
'IMU STABILIZATION IS STRAPDOWN'
ININIT
ATUDE
INITIALIZE ATTITUDE PREDICTOR FILTER
RETURN

2.3.18 Function & Equations

A) The purpose of SDATUD is to form the "incremental gimbal angles" that would be produced by a strapdown IMU. These three angles are incremental "body" rotations (as opposed to incremental Euler angles), that is, they are the projection, onto the body frame axes, of $\underline{r}$ (where $|\underline{r}|$ is the angle of rotation about a unit vector $\frac{\underline{r}}{|\underline{r}|}$ or $\underline{r}$ is the rotation implicit in going from the old CO$5 matrix to the new CO$5 matrix). The incremental angles are then quantized, and the residuals saved for next cycle.

B) i) The rotation (in DCM form) is given as

$$R = (CO\$5OLD)^{-1} (CO\$5NEW)$$

ii) The rotation DCM generated by a rotation of Θ about a unit vector $\underline{n}$ (components n_1, n_2, n_3) is given analytically as

$$R = (1 - \cos\Theta) \begin{bmatrix} n_1 n_1 & n_1 n_2 & n_1 n_3 \\ n_2 n_1 & n_2 n_2 & n_2 n_3 \\ n_3 n_1 & n_3 n_2 & n_3 n_3 \end{bmatrix} + \cos\Theta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$+ \sin\Theta \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

since first two matrices are symmetric

$$.5 * (R(3,2) - R(2,3)) = \sin\Theta \; n_1 \quad (\text{call this } \rho_1)$$

$$.5 * (R(1,3) - R(3,1)) = \sin\Theta \; n_2 \quad (\text{call this } \rho_2)$$

$$.5 * (R(2,1) - R(1,2)) = \sin\Theta \; n_3 \quad (\text{call this } \rho_3)$$

Now $\rho_1^2 + \rho_2^2 + \rho_3^2 = (\sin\theta)^2 (n_1^2 + n_2^2 + n_3^2) = (\sin\theta)^2$ since $\underline{n}$ is unit vector.

iii) One can then find $\sin\theta$, θ, and n_1, n_2, n_3 and compute θn_1, θn_2, and θn_3 which are the required results.

C) C0$5 is found by computing C0$2P (stored in scratch matrix) and multiplying by C0P$5 (which was computed by SSINTG and stored in C2P$5T.

(D) Once the incremental angles are computed, they must be quantized (if necessary). With $\underline{R}^*$ representing the residuals from the previous cycle, and $[\underline{X}]$ implying that each component is to be rounded down to an integral number of units.

$$\underline{TEMP} = [\theta\underline{n} + \underline{R}^*]$$

$$R = \theta\underline{n} + \underline{R}^* - \underline{TEMP}$$

$$\theta\underline{n} = \underline{TEMP}$$

E) $(CO\$5OLD)^{-1}$ is assumed equal to the transpose of CO$5OLD.

Inputs, Outputs & Internal Variables

LONG$T - longitude per PROFGEN
LONG0 - initial PROFGEN longitude
TIME - current simulated time
TIME0 - initial simulated time
WERT - earth rate
SLAT$T - sine of PROFGEN latitude
CLAT$T - cosine of PROFGEN latitude
SALF$T - sine of PROFGEN alpha
CALF$T - cosine of PROFGEN alpha

} these inputs used in computation of C0$2P

C0$5 - on input, previous value of C0$5
on output next value of C0$5

ETA$P(1) - X rotation
ETA$P(2) - Y rotation
ETA$P(3) - Z rotation

} outputs

STH - internal variable used for sinθ above

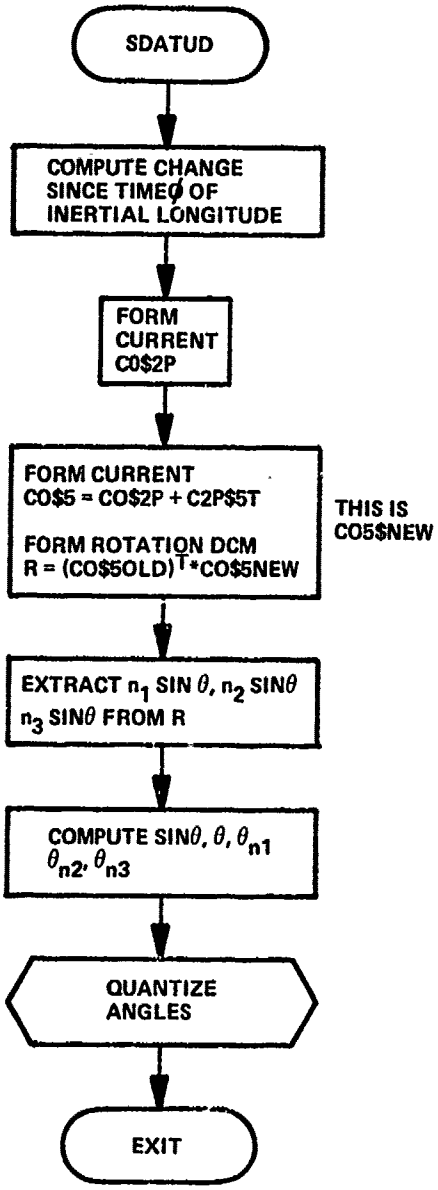
TH - internal variable used for θ above

A matrix- internal variable used for R above

B matrix- internal variable used for C0$5NEW above, until placed in C0$5

SF - internal variable used for TH/STH

TMP vector - internal variable; TMP(1) is used to hold change in longitude from initial inertial position


SDATUD
COMPUTE CHANGE SINCE TIME0 OF INERTIAL LONGITUDE
FORM CURRENT C0$2P
FORM CURRENT CO$5 = CO$2P + C2P$5T
FORM ROTATION DCM R = (CO$5OLD)T*CO$5NEW
THIS IS CO5$NEW
EXTRACT n1 SIN θ, n2 SINθ n3 SINθ FROM R
COMPUTE SINθ, θ, θn1 θn2, θn3
QUANTIZE ANGLES
EXIT

SDINTG

2.3.19 Function & Equations

The purpose of SDINTG is to take PROFGENs specific forces (as coordinatized in PROFGENs platform frame) and rotate them into the body frame. Mathematically

$$\underline{f}^5 = C^5_{2p}\, C^{2p}_{2QP}\, \underline{f}^{2QP}$$

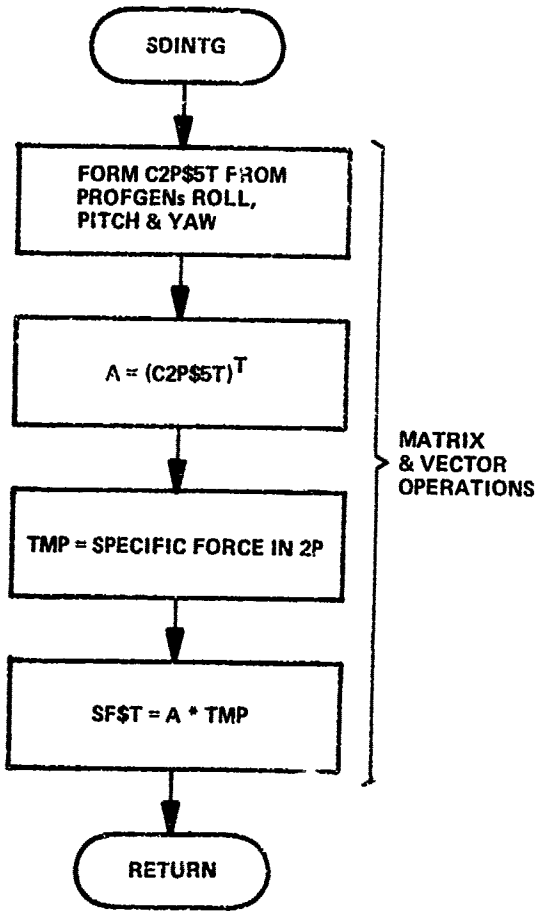
(where 2QP is PROFGENS North-West-Up platform frame). The C^5_{2p} is formed from PROFGENS roll, pitch and yaw.

Inputs, Outputs & Internal Variables

Inputs ETA$T - roll, pitch & yaw

Inputs & output SF$T - specific force: input in 2QP, output in 5

Scratch variables - vector TMP (internal) (used to hold $C^{2p}_{2QP}\underline{f}^{2QP}$), array A (used to hold C^5_{2p})


SDINTG
FORM C2P$5T FROM PROFGENs ROLL, PITCH & YAW
A = (C2P$5T)^T
TMP = SPECIFIC FORCE IN 2P
SF$T = A * TMP
RETURN
MATRIX & VECTOR OPERATIONS

HSINTG

2.3.20 Function & Equations

A) The purpose of HSINTG is to perform the calculations associated with the strapdown high speed delta -V integration that would be implemented in an avionics computer. This integration takes the delta -V's from the accelerometers (at the attitude cycle rate), transforms them from the frame of the accelerometers (synonymous with the body frame) to the computational frame, and sums them over the navigation cycle. The output of this routine is therefore the delta -V, over the navigation cycle, in the computational frame. This subroutine also must update C_5^{2P} at the attitude rate.

B) The exact equation relating accelerations in the two frames is:

$$\underline{a}^{2P} = C_5^{2P}\underline{a}^5$$

so that

$$\underline{\Delta V}^{2P} = \int_t^{t+DELT} \underline{a}^{2P}\, dt = \int_t^{t+DELT} C_5^{2P}\, \underline{a}^5 \quad dt$$

where DELT is the navigation computations cycle. Therefore

$$\underline{\Delta V}^{2P} = \sum_{i=1}^{n} \int_{t+(i-1)DELTA}^{t+iDELTA} C_5^{2P}\, \underline{a}^5 \quad dt$$

where DELTA is the attitude cyle and

$$n = DELT/DELTA.$$

Assuming the $\underline{a}^5$ to be constant over an attitude cycle, it can be shown (see the appendices) that the following approximation is good to $0(\Delta\theta^2)$.

$$\underline{\Delta V}^{2P} = \sum_{i=1}^{n} \left[\left\{ \left[C_5^{2P}(t + (i-1)\Delta t) \right] + \left[C_5^{2P}(t+i\ \Delta t) \right] \right\} \underline{a}^5/2 \right]$$

where $C_5^{2P}(X)$ is value of C_5^{2P} at time X

C) The exact differential equation defining the time rate of change of C_5^{2P} is

$$\dot{C}_5^{2P} = C_5^{2P} \times \underline{\omega}_{2P,5}^5$$

where $\underline{\omega}_{2P,5}^5$ is the angular rate of the 2P frame with respect to the 5 frame, as expressed in the 5 frame. Now

$$\underline{\omega}_{2P,5}^5 = \underline{\omega}_{o,5}^5 - \underline{\omega}_{o,2P}^5$$

$$= \underline{\omega}_{o,5}^5 - C_{2P}^5\ \underline{\omega}_{o,2P}^{2P}$$

The $\underline{\omega}_{c,5}^5$ (or, more accurately, DELTA $\underline{\omega}_{o,5}^5$) is the physical data measured by the strapdown IMU's gyros. $\underline{\omega}_{o,2P}^{2P}$ is computed by the navigation equations.

D) HSINTG can be used either of two methods to compute the value of C2P$5, as follows:

i) If IPC(27) equals zero, the new value of C2P$5 is computed by using the DCM update routine, DCMUPD. Orthonomalization of C2P$5 is performed every HSOINC seconds, beginning with the initial value of HSOTIM. The order of the update is given by IPC(26) plus one.

ii) If IPC(27) equals one, the new value of C2P$5 is computed by using a quaternion scheme, where the quaternion q_5^{2p} (stored as Q2P$5) represents the rotation from the 5 frame to the 2P frame. Specifically:

a) The quaternion corresponding to the rotation of the 2P frame during the attitude cycle p_{2p}^{2p*} (stored as P0, P1, P2, P3) is formed to first, second or third order, depending on the value of IPC(26).

The following formulae are derived in an appendix:

1) First order

$$P = [1, \Delta X/2, \Delta Y/2, \Delta Z/2]$$

where ΔX, ΔY, ΔZ are the X, Y and Z components of the rotation (see SDATUD documentation)

2) Second order

$$P = \left[1 - \frac{\left(\frac{\Delta X}{2}\right)^2 + \left(\frac{\Delta Y}{2}\right)^2 + \left(\frac{\Delta Z}{2}\right)^2}{2}, \frac{\Delta X}{2}, \frac{\Delta Y}{2}, \frac{\Delta Z}{2}\right]$$

3) Third order

$$P = \left[1 - \frac{\left(\frac{\Delta X}{2}\right)^2 + \left(\frac{\Delta Y}{2}\right)^2 + \left(\frac{\Delta Z}{2}\right)^2}{2}, \text{CON} * \frac{\Delta X}{2} + \text{XDTX}, \text{CON} * \frac{\Delta Y}{2} + \text{XDTY}, \text{CON} * \frac{\Delta Z}{2} + \text{XDTZ}\right]$$

where

$$CON = 1 - \frac{1}{6}\left[\left(\frac{\Delta X}{2}\right)^2 + \left(\frac{\Delta Y}{2}\right)^2 + \left(\frac{\Delta Z}{2}\right)^2\right]$$

$$XDTX = \frac{1}{6}\left(\frac{\Delta Z \Delta Y^*}{2} - \frac{\Delta Z^* \Delta Y}{2}\right)$$

$$XDTY = \frac{1}{6}\left(\frac{\Delta X \Delta Z^*}{2} - \frac{\Delta X^* \Delta Z}{2}\right)$$

$$XDTZ = \frac{1}{6}\left(\frac{\Delta X \Delta Y^*}{2} - \frac{\Delta X^* \Delta Y}{2}\right)$$

ΔX^*, ΔY^* and ΔZ^* are previous values of ΔX, ΔY and ΔZ.

However, these XDTX, XDTY and XDTZ terms are the result of a $\dot{\Omega} \times \Omega$ term (see DCMUPD). Due to the infinite angular accelerations of PROFGEN, XDTX, XDTY and XDTZ have beer set equal to zero.

b) The quaternion is updated by

$$q_5^{2p^*} = q_5^{2p} \times p_{2p}^{2p^*} \quad \text{where}$$

* indicates "end of attitude cycle" (new) value and 'X' indicates quaternion muliplication.

Therefore (dropping frames from notation and indicating quaternion components with subscripts):

$$\begin{bmatrix} q_0^* \\ q_1^* \\ q_2^* \\ q_3^* \end{bmatrix} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

c) The quaternion is normalized every HSOINC seconds starting with the time indicated by the initial value of HSOTIM.

A quaternion is normal if the sum of the squares, of its components, is equal to 1. We therefore want to multiply each component by γ, so that

$$(\gamma q_0)^2 + (\gamma q_1)^2 + (\gamma q_2)^2 + (\gamma q_3)^2 = 1$$

This imples

$$\gamma = \frac{1}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} =$$

$$\left(1 + (q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1)\right)^{-\frac{1}{2}}$$

A first order approximation (that is, assuming $q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1$ is small) to γ is

$$1 - \frac{1}{2}(q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1)$$

$$= 1.5 - .5^*(q_0^2 + q_1^2 + q_2^2 + q_3^2)$$

d) $C_5^{2p^*}$ is constructed from $q_5^{2p^*}$ according as follows:

$$C_5^{2p^*} = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_3q_1 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

e) After the new C2P$5 has been computed the average of the new C2P$5 and the old (that is, from the previous attitude cycle) is computed. The delta -V's in the body frame are pre-multiplied by this average, yielding the delta -V (for the current attitude cycle) in the computational frame. This quantity is then added in to the total for the current navigation cycled (stored in DV5$2P).

Inputs, Outputs & Internal Variables

Inputs:

C2P$5 - previous value of C2P$5

THTTRQ - $\underline{\omega}^{2p}_{0,2p}$ * DELT; DELT is nav cycle; $\omega^{2p}_{0,2p}$ is rate of comp frame with respect to inertial, as computed by INRNAV

DELT - Nav cycle

DELTA - attitude cycle

Q2P$5 - previous value of quaternion

HS0TIM - next time to perform normalization (orthonormalization)

HS0INC - time between normalizations (orthonomalizations)

ETA$P - incremental angles sensed by strapdown gyros during an attitude cycle

ODX, ODY, ODZ - previous values of $\underline{\omega}^{5}_{2p,5}$ * DELTA

Outputs:

C2P$5 - current value of C2P$5

ODX, ODY, ODZ - "previous values" for next attitude cycle

Q2P$5 - current value of rotation quaternion

Internal Variables:

TMP $\underline{\omega}^{5}_{2P,5}$ DELTA rotation of 2P with respect to 5 frame over one attitude cycle

P0, P1, P2, P3 — components of quaternion representing rotation of 2P frame during an attitude cycle

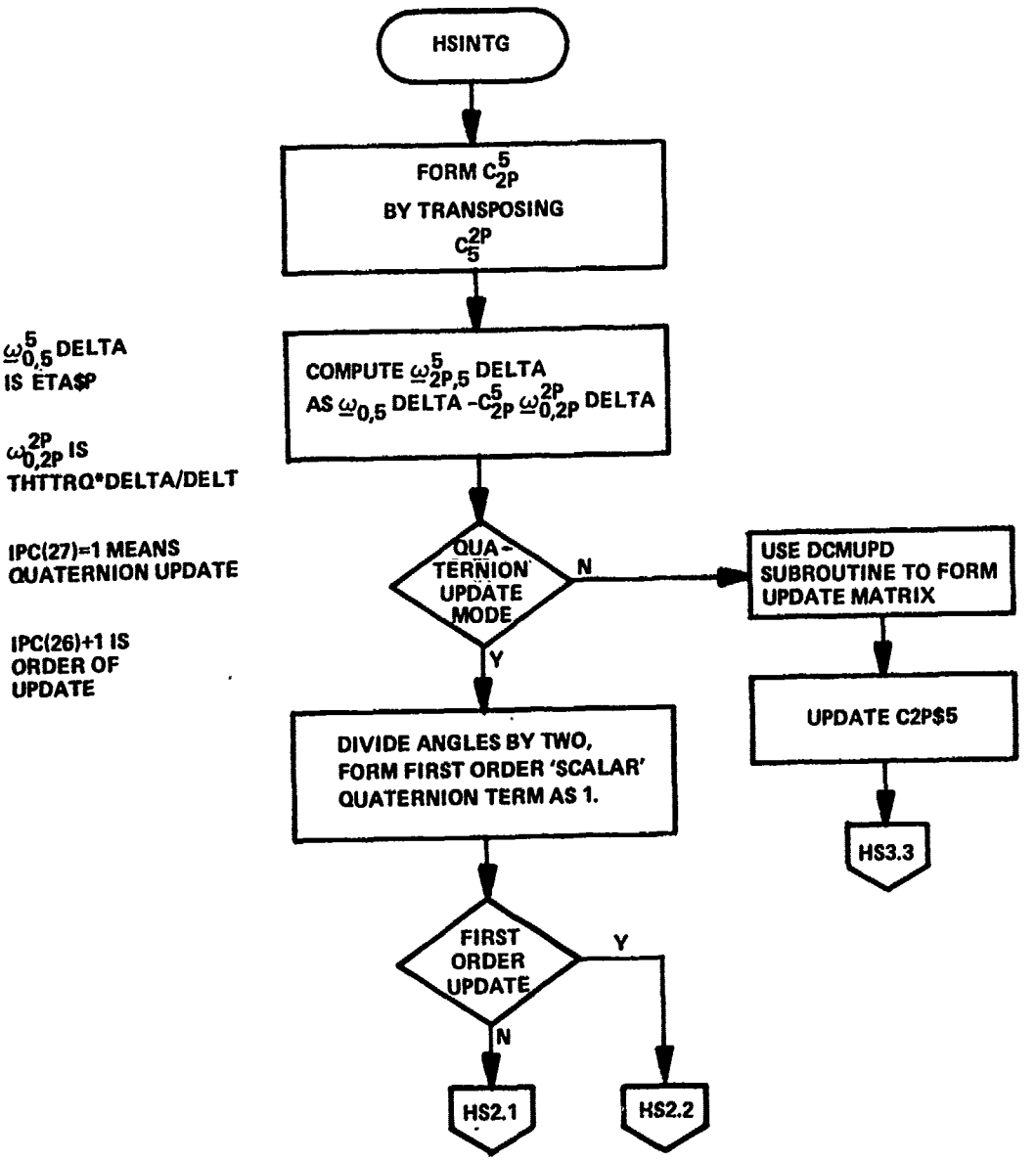
CON, TS02, COR — internal scratch variables

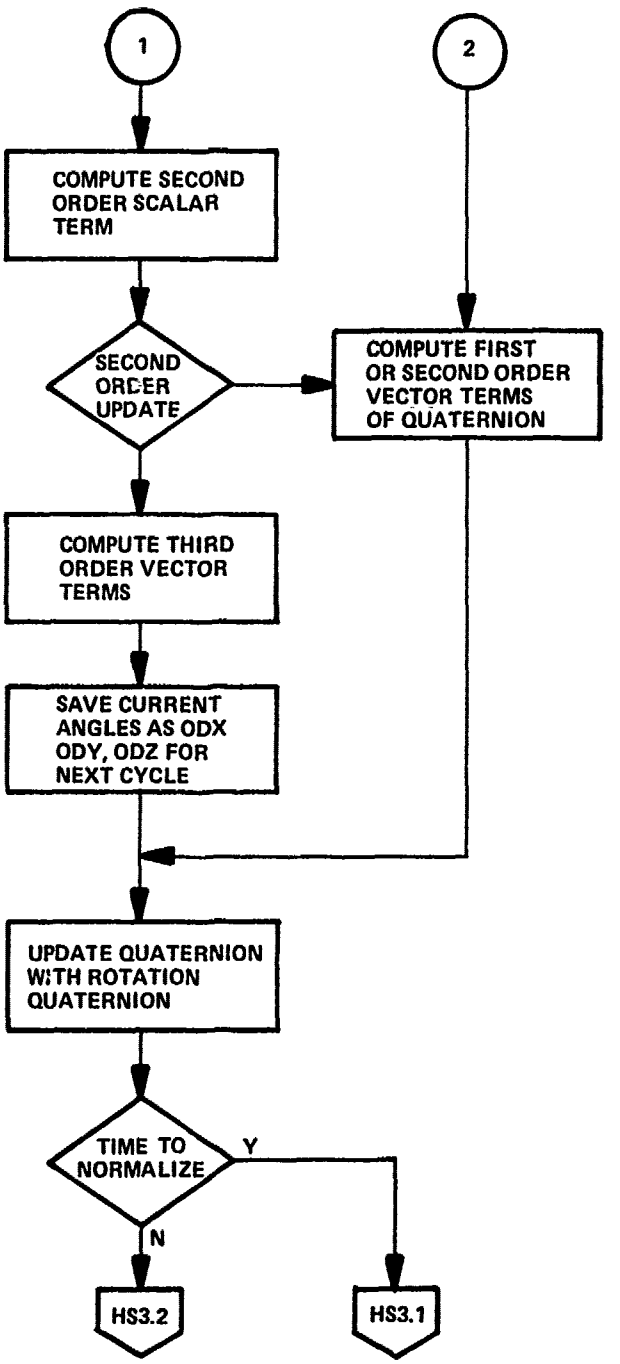
Q0, Q1, Q2, Q3 — EQUIVALENCEd to Q2P$5

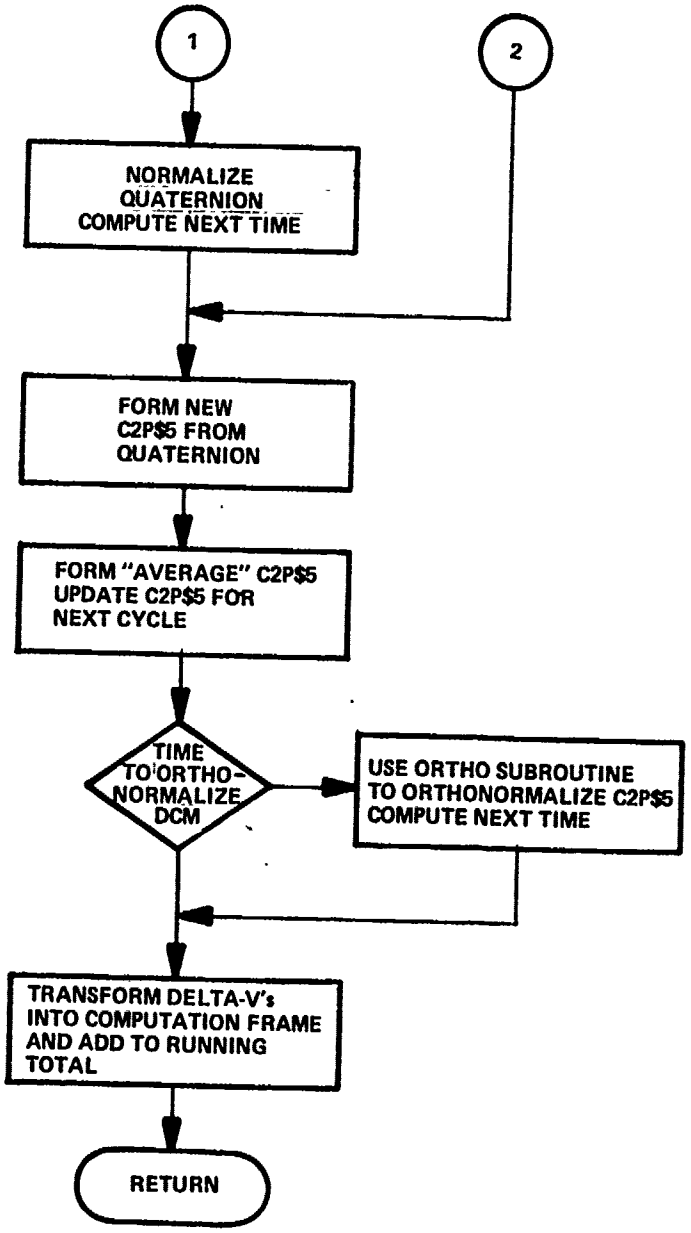
XDTX, XDTY, XDTZ — Elements of third order quaternion rotating to $\dot{\Omega} \times \Omega$

Q0N, Q1N, Q2N, Q1S, Q2S, Q3S, Q01, Q02, Q03, Q12, Q23, Q31 — internal scratch variables

```
B       holds new value of C2P$5 until averaging is through
A       holds average of new and old value of C2P$5
```


HSINTG
FORM C_{2P}^{5} BY TRANSPOSING C_{5}^{2P}
COMPUTE $\omega_{2P,5}^{5}$ DELTA AS $\omega_{0,5}$ DELTA $-C_{2P}^{5}\,\omega_{0,2P}^{2P}$ DELTA
QUA-TERNION UPDATE MODE
N
Y
USE DCMUPD SUBROUTINE TO FORM UPDATE MATRIX
UPDATE C2P$5
HS3.3
DIVIDE ANGLES BY TWO, FORM FIRST ORDER 'SCALAR' QUATERNION TERM AS 1.
FIRST ORDER UPDATE
Y
N
HS2.1
HS2.2
$\omega_{0,5}^{5}$ DELTA IS ETA$P
$\omega_{0,2P}^{2P}$ IS THTTRQ*DELTA/DELT
IPC(27)=1 MEANS QUATERNION UPDATE
IPC(26)+1 IS ORDER OF UPDATE


1
2
COMPUTE SECOND ORDER SCALAR TERM
SECOND ORDER UPDATE
COMPUTE FIRST OR SECOND ORDER VECTOR TERMS OF QUATERNION
COMPUTE THIRD ORDER VECTOR TERMS
SAVE CURRENT ANGLES AS ODX ODY, ODZ FOR NEXT CYCLE
UPDATE QUATERNION WITH ROTATION QUATERNION
TIME TO NORMALIZE
Y
N
HS3.2
HS3.1


1
2
NORMALIZE QUATERNION COMPUTE NEXT TIME
FORM NEW C2P$5 FROM QUATERNION
FORM "AVERAGE" C2P$5 UPDATE C2P$5 FOR NEXT CYCLE
TIME TO ORTHO-NORMALIZE DCM
USE ORTHO SUBROUTINE TO ORTHONORMALIZE C2P$5 COMPUTE NEXT TIME
TRANSFORM DELTA-V's INTO COMPUTATION FRAME AND ADD TO RUNNING TOTAL
RETURN

HSINT2

2.3.21 Function & Equations

The purpose of HSINT2 is to coordinate the transmission of the delta-V information from the high speed strapdown delta-V summation into the nav equations. It is called just before INRNAV, at which time it picks up the delta-V from DV5$2P (where HSINTG is accumulating them) places it in DV$P (from which INRNAV will input them) and then clears DV5$2P.

Input, Outputs & Internal Variables

Input & Outputs: DV5$2P

Output: DV$P

QNTIZ

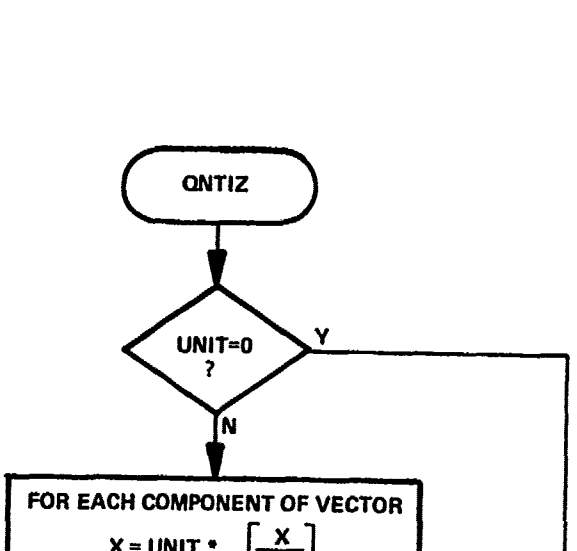
2.3.22 Function & Equations

The purpose of QNTIZ is to perform the quantization function (eg. an accelerometer may generate delta -V's as multiples of 10^{-6} g-sec, or a gimbal angle resolver which generates angles as an integral number of arc minutes). The input parameters to the routine is the vector to be quantized. If the unit is zero, no quantization is done, else (for each component of the vector):

$$X = UNIT * \left[\frac{X}{UNIT}\right]$$ where [y] is the 'greatest integer in' y (that is, value rounded down).

Inputs, Outputs & Internal Variables

- ARRAY - parameter representing input vector; output values also returned here
- UNIT - parameter representing unit in which to quantize
- I - internal scratch variable


QNTIZ
UNIT=0
?
Y
N
FOR EACH COMPONENT OF VECTOR
X = UNIT * [X/UNIT]
RETURN

ORTHO

2.3.23 Function & Equations

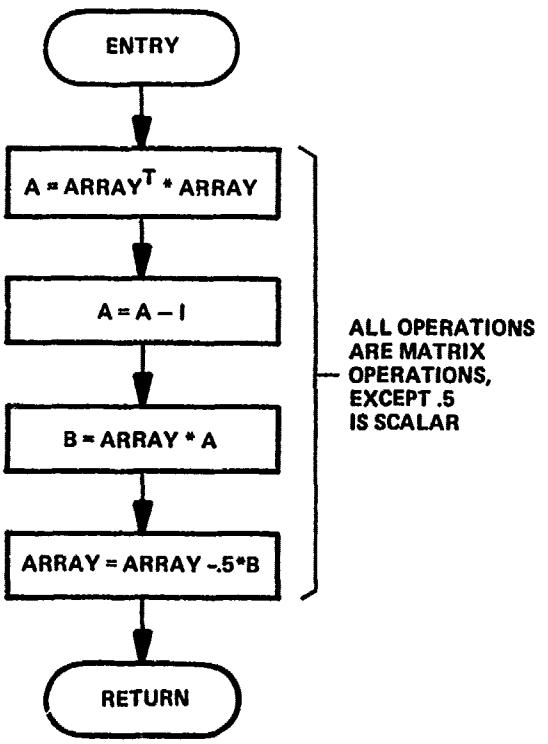
A) The purpose of ORTHO is to orthonomalize, to first order, a 3 x 3 array passed to it as input. The update used is.

$$ARRAY \leftarrow ARRAY - .5 * ARRAY * (ARRAY^T * ARRAY - I)$$

where I is the identity matrix, ARRAYis the ARRAY to be updated, and $ARRAY^T$ is the transpose of ARRAY.

Inputs, Outputs & Internal Variables

ARRAY - parameter representing input argument
I, J - index scratch
A, B - scratch 3 x 3 matrices


ENTRY
A = ARRAY^T * ARRAY
A = A – I
B = ARRAY * A
ARRAY = ARRAY –.5*B
RETURN
ALL OPERATIONS ARE MATRIX OPERATIONS, EXCEPT .5 IS SCALAR

MATVEC

2.3.24 Name

Matrix Times Vector (MATVEC) routine.

Function

This subroutine multiplies any 3 x 1 vector by any 3 x 3 matrix.

Operation and Equations

Calling sequence is given below:

```
CALL MATVEC (D, B, C)
```

where:

D = 3 x 1 vector result
B = 3 x 3 matrix
C = 3 x 1 input vector

$\overline{D} = [B]\ \overline{C}$

Only vector D is modified upon return to the caller.

Input Variables

Defined in calling statement.

Internal Variables

A - 3 x 1 vector to collect results of multiplications.

Output Variables

Defined in calling statement.

MTM

2.3.25 Name

Matrix Transpose Multiply (MTM) routine.

Function

This subroutine multiplies any 3x3 matrix by the transpose of any 3x3 matrix.

Operation and Equations

Calling sequence is given below:

CALL MTM(A,B,C)

where:

$$[A] = [B]^T \quad [C]$$

The same matrix can be used for more than one argument. Only matrix A is modified upon return to the caller. The detailed equation is:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \times \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

Input Variables

Defined in the calling statement.

Internal Variables

AA - Used to collect result of matrix multiply.

Output Variables

Defined in the calling statement.

2.3.26 Name

Matrix Multiply (MM) routine.

Function

Multiply any 3x3 matrix by any 3x3 matrix.

Operation and Equations

Calling sequence is given below:

CALL MM(D,B,C)

where:

[D] = [B][C]

The same matrix can be used for more than one argument. Only matrix D is modified upon return to the calling program.

Input Variables

Defined in calling statement.

Internal Variables

A - Used to collect result of matrix multiply.

Output Variables

Defined in calling statement.

2.3.27 Name

XYZ Rotation Matrix (ROTXYZ) routine.

Function

Forms the 3x3 rotation matrix using three Euler rotations.

Operation and Equations

Calling sequence is:

CALL ROTXYZ (ARRAY, SZ, CZ, SY, CY, SX, CX)

where:

ARRAY = 3x3 rotation matrix returned to calling routine
SZ = sine of Z rotation
CZ = cosine of Z rotation
SY = sine of Y rotation
CY = cosine of Y rotation
SX = sine of X rotation
CX = cosine of X rotation

The sequence of rotations is minus Z degrees about Z, minus Y degrees about Y, and X degrees about X. The detailed matrix equation is given below:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos X & \sin X \\ 0 & -\sin X & \cos X \end{bmatrix} \begin{bmatrix} \cos Y & 0 & \sin Y \\ 0 & 1 & 0 \\ -\sin Y & 0 & \cos Y \end{bmatrix} \begin{bmatrix} \cos Z & -\sin Z & 0 \\ \sin Z & \cos Z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where the a matrix is returned to the caller as the variable ARRAY.

Input Variables

Defined in calling sequence.

Internal Variables

SYCZ = SY*CZ
SYSZ = SY*SZ
where SY, SZ, and CZ are defined in the call statement

Output Variables

Defined in calling sequence.

2.3.28 Name

Matrix Transpose (MATRAN) routine.

Function

Transpose any 3x3 matrix.

Operation and Function

Calling sequence is given below:

CAL MATRAN (B,A)

where:

$[B] = [A]^T$

A and B must be two different 3x3 matrices.

Input Variables

Defined in calling sequence

Internal Variables

None.

Output Variables

Defined in calling sequence.

2.3.2.29 Name

ZYX Rotation Matrix (ROTZYX) routine.

Function

Forms a 3x3 rotation matrix using 3 Euler rotations. Sequence of rotations is X degrees about X, Y degrees about Y, and Z degrees about Z.

Operation and Equations

Calling sequence is given below:

CALL ROTZYX (ARRAY, SZ, CZ, SY, CY, SX, CX)

where:

ARRAY - 3x3 rotation matrix returned to calling routine
SZ - sine of Z rotation
CZ - cosine of Z rotation
SY - sine of Y rotation
CY - cosine of Y rotation
SX - sine of X rotation
CX - cosine of X rotation

The detailed matrix equation is given below:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \cos Z & \sin Z & 0 \\ -\sin Z & \cos Z & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos Y & 0 & -\sin Y \\ 0 & 1 & 0 \\ \sin Y & 0 & \cos Y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos X & \sin X \\ 0 & -\sin X & \cos X \end{bmatrix}$$

Where the a matrix is returned to the caller as the variable ARRAY.

Input Variables

Defined in the calling sequence.

Internal Variables

SYSX = SY*SX
SYCX = SY*CX

Output Variables

Defined in the calling sequence.

2.3.30 Function & Equations

The purpose of DCMUPD is to perform as a utility subroutine which can form up a matrix representation of a "small" rotation of a "DCM" (Direction Cosine Matrix). DCMUPD can make a first, second, or third order approximation to the exact form of the update matrix (see the accompanying analytic discussion for an explanation of 'order' and the derivation of the formulas used by this subroutine). DCMUPD can be called in such a way that the resulting output is either

a) A rotation matrix, that is the updated DCM can be formed by taking the old DCM and post multiplying the output of DCMUPD (the diagonal entries of such a matrix are 'nominally' 1)

b) A matrix which, when pre-muliplied by the old DCM, yields the change (in the additive sense) to the old DCM. (the diagonal entries of such a matrix are 'nominally' 0)

The inputs to DCMUPD are

i) DX, DY, DZ - components of rotation vector, that is $\underline{\gamma} = DX\hat{}_x + DY\hat{}_y + DZ\hat{}_z$ represents a rotation of $|\gamma|$ radians about the unit vector which is $\frac{\underline{\gamma}}{|\underline{\gamma}|}$.

ii) DIAG the 'nominal' value of the diagonal entries

iii) IORDM1 - one less than the order of the update

iv) ODX, ODY, ODZ - previous values of ODX, ODY and ODZ* (see footnote)

For a first order update, the following formula is used (DCM is the output of the routine).

$$DCM = \begin{bmatrix} DIAG & -DZ & DY \\ DZ & DIAG & -DX \\ -DY & DX & DIAG \end{bmatrix}$$

For a second order update, the following matrix is added on to the first order results

$$\begin{bmatrix} DY^{**}2 - DZ^{**}2 & DX*DY & DZ*DX \\ DX*DY & -DZ^{**}2 - DX^{**}2 & DY*DZ \\ DZ*DX & DY*DZ & -DX^{**}2 - DY^{**}2 \end{bmatrix}$$

For a third order update, the following matrix is added to the sum of the first and second order matrices*

$$\begin{bmatrix} 0 & -2*Q*DZ \\ & -DY*ODX \\ & +DX*ODY \end{bmatrix} \quad \begin{bmatrix} -2*Q*DY \\ -DZ*ODX \\ +DX*ODZ \end{bmatrix}$$

$$\begin{bmatrix} -2*Q*DZ \\ +DY*ODX \\ -DX*ODY \end{bmatrix} \quad \begin{bmatrix} 0 \end{bmatrix} \quad \begin{bmatrix} -2*Q*DX \\ -DZ*ODY \\ +ODY*DZ \end{bmatrix}$$

$$\begin{bmatrix} -2*Q*DY \\ +DZ*ODX \\ -DX*ODZ \end{bmatrix} \quad \begin{bmatrix} -2*Q*DX \\ +DZ*ODY \\ -DY*ODZ \end{bmatrix} \quad \begin{bmatrix} 0 \end{bmatrix}$$

where Q = 2 + (DX**2 + DY**2 + DZ**2)

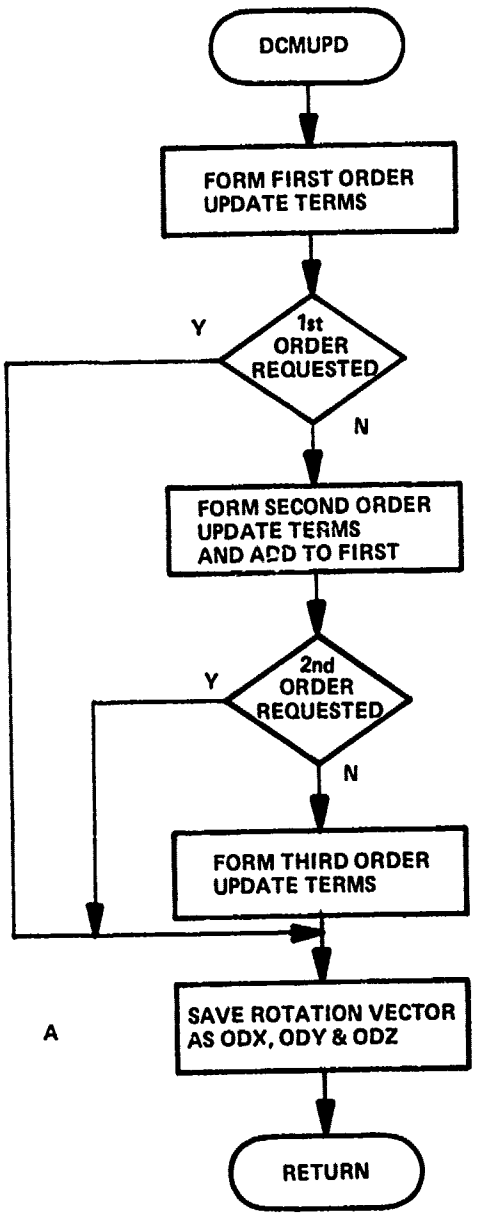
* The terms including ODX, ODY and ODZ are part of the third order update only and arise from an angular acceleration term. Due to the step changes in PROFGEN's angular rates, these terms have been omitted from the subroutine.

Inputs, Outputs & Internal Variables

The inputs were described above.

The outputs are DCM (the DCM update matrix) and ODX ODY, & ODZ, which are set equal to DX, DY and DZ (respectively) just prior to exit and are to be used during the next update of the DCM.

Internal variables S1, S2, S3, T1, T2, T3 and SCL.


DCMUPD
FORM FIRST ORDER UPDATE TERMS
Y
1st ORDER REQUESTED
N
FORM SECOND ORDER UPDATE TERMS AND ADD TO FIRST
Y
2nd ORDER REQUESTED
N
FORM THIRD ORDER UPDATE TERMS
A
SAVE ROTATION VECTOR AS ODX, ODY & ODZ
RETURN

LLINTG

2.3.31 Function & Equations

The purpose of LLINTG is to transform the specific forces from the PROFGEN platform frame to the local-level platform frame. If the effects of torquing are not considered, the transformation is given below

$$\underline{f}^{2'} = C^{2'}_{2Q'}\, \underline{f}^{2Q'}$$

2Q' denotes PROFGEN platform frame

$$C^{2'}_{2Q'} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2' denotes local-level wander azimuth

When torquing commands from the navigation equations are introduced, the platform can become misaligned from the true local-level wander azimuth frame. These misalignments must be found and used when calculating the specific force sensed by the platform. The necessary calculations are as follows:

1. Transform velocity from PROFGEN platform (2Q') to local vertical north (2).

$$\underline{V}^2 = C^2_{2Q'}\, \underline{V}^{2Q'}$$

$$C^2_{2Q'} = \begin{bmatrix} 0 & 0 & 1 \\ -\sin\alpha_T & -\cos\alpha_T & 0 \\ \cos\alpha_T & -\sin\alpha_T & 0 \end{bmatrix}$$

α_T - PROFGEN wander angle

2. Calculate radii or curvature

$$r_m = h + \frac{r_e(1-\varepsilon^2)}{(1-\varepsilon^2 \sin^2 L)^{3/2}}$$

$$r_p = h + \frac{r_e}{(1 - \varepsilon^2 \sin^2 L)^{1/2}}$$

where:

r_m - Meridian radius

r_p - Prime vertical radius

h - Altitude above sea level

r_e - Sea level equatorial radius

ε - Earth eccentricity

L - Geodetic latitude

3. Compute the angular rates due to the velocity and earth rate then transform to actual platform frame

$$\omega_U = \omega_e \sin L \text{ (rotation about vertical)}$$

$$\omega_E = -V_n / r_m \text{ (rotation about east)}$$

$$\omega_N = V_e / r_p + \omega_e \cos L \text{ (rotation about north)}$$

$$\omega_T = C_{2'}^{2M'} \; C_2^{2'} \; \omega$$

where:

ω_e - Earth angular rate

ω - Angular rate in local vertical north frame $= \begin{bmatrix} \omega_U \\ \omega_E \\ \omega_N \end{bmatrix}$

V_n - Aircraft velocity in north direction

V_e - Aircraft velocity in east direction

$$C_2^{2'} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_T & \sin\alpha_T \\ 0 & -\sin\alpha_T & \cos\alpha_T \end{bmatrix}$$

$C_{2'}^{2M'}$ = Transformation from LLWA frame (2') to the misaligned LLWA frame (2M') calculated on previous cycle (see below).

ω_T = Angular rate in 2M' frame due to trajectory velocities.

4. Update the misalignment matrix ($C_{2'}^{2M'}$). The torque commands are quantized in order to represent the resolution of an actual torquer. The commands are then combined with the angular rates calculated in step 3 to give the angular error due to the difference between commanded orientation and LLWA orientation. These misalignment angles are put into a skew symmetric matrix which is used to update the misalignment matrix.

$$\theta_E = \theta_{QT} - \theta_T$$

$$C_{2'}^{2M'} = AC_{2'}^{2M'}$$

where:

θ_E - Angular error

θ_{QT} - Angle traversed by IMU during one sample time due to quantized commanded torque

θ_T - Angle traversed by IMU during one sample time due to aircraft's velocity over earth

$$A = \begin{bmatrix} 1 & \theta_{E3} & -\theta_{E2} \\ -\theta_{E3} & 1 & \theta_{E1} \\ \theta_{E2} & -\theta_{E1} & 1 \end{bmatrix}$$

4. Specific force is transformed from the PROFGEN platform frame (2Q') to the misaligned LLWA frame (2M')

$$\underline{f}^{2'} = C_{2'}^{2M'} C_{2Q'}^{2'} \underline{f}^{2Q'}$$

f - Specific force in indicated frame.

Input Variables

CALF$T - Cosine of wander angle from PROFGEN.

SALF$T - Sine of wander angle from PROFGEN.

V$T - Velocity vector from PROFGEN in north, west, up frame.

ESQ - Earth eccentricity squared.

SLAT$T - Sine of geodetic latitude from PROFGEN

HB - Altitude from PROFGEN

R0 - Earth equatorial radius

RESQ - R0*ESQ

THTTRQ - Commanded torque vector: angle to be traversed during the navigation cycle.

TQUANT - Torquing angle quantization.

DELT - Navigation cycle period.

IPC(1) - Flag for operator controlled printout of the angular rate due to aircraft velocity (WT). If 0 no printout. Default is 0.

IPC(13)- Flag for operator controlled printout of misalignment angles. If 0 no printout. Default is 0.

SF$T - Specific force vector from PROFGEN in. North, west, up frame.

IOPNLP - Flag to control whether or not misalignments due to torquing are to be calculated. 0 mean calculate misalignments. Default is 0.

Internal Variables

VN - Aircraft velocity in north direction.

VE - Aircraft velocity in east direction.

TMP1, TMP2 - Scratch variables for calculating radii or curvature.

RP - Prime vertical radius of curvature.

RM - Meridian radius of curvature.

WT - Angular rate of aircraft due to aircraft velocity over the earth in LLWA frame.

WTE - Angular rate of aircraft about east.

WTN - Angular rate of aircraft about north.

TRESID - Residual torque angle generated as a result of quantizing the torque command.

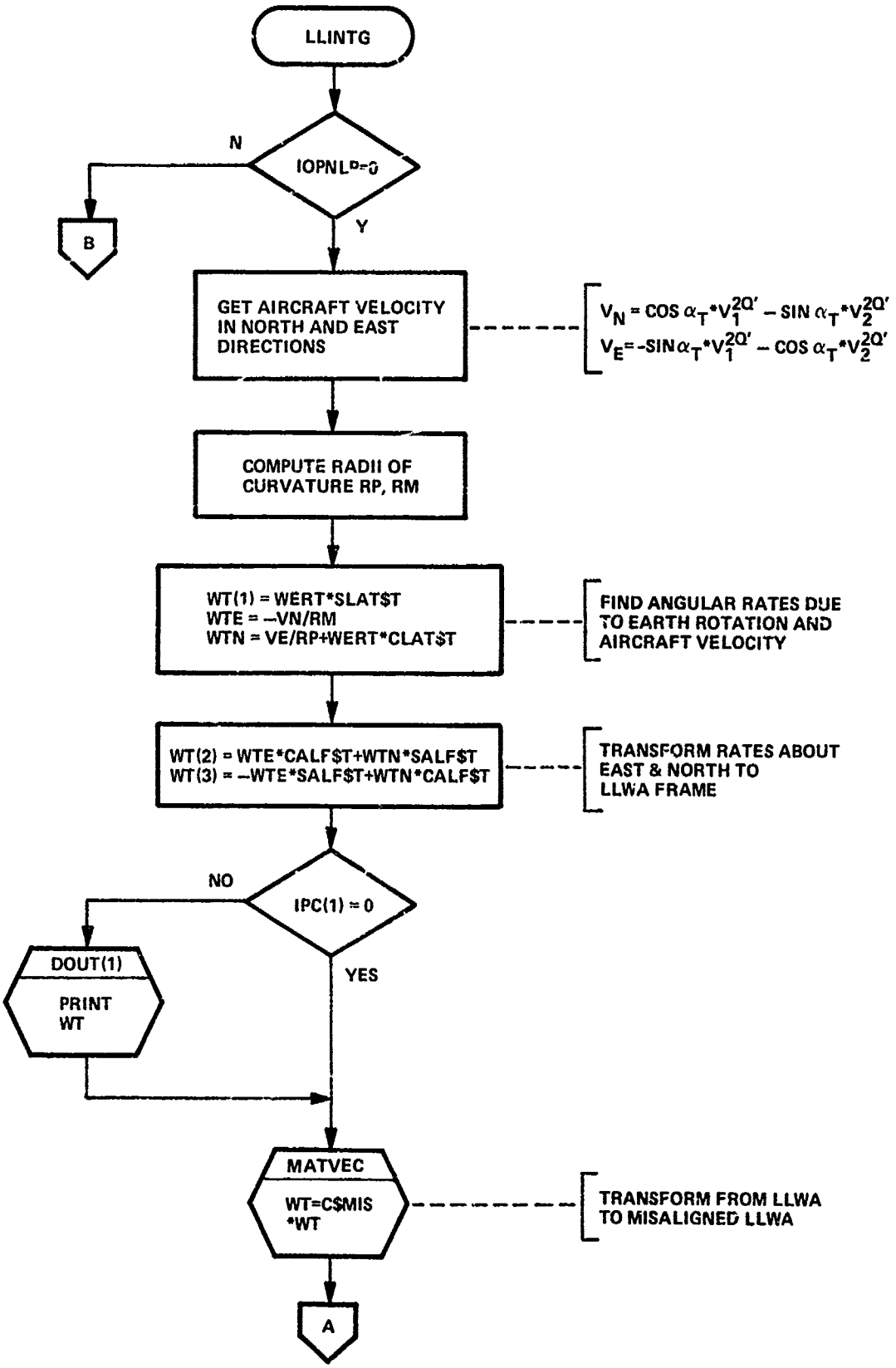
A - 3x3 skew symmetric matrix representing the incremental misalignment angles

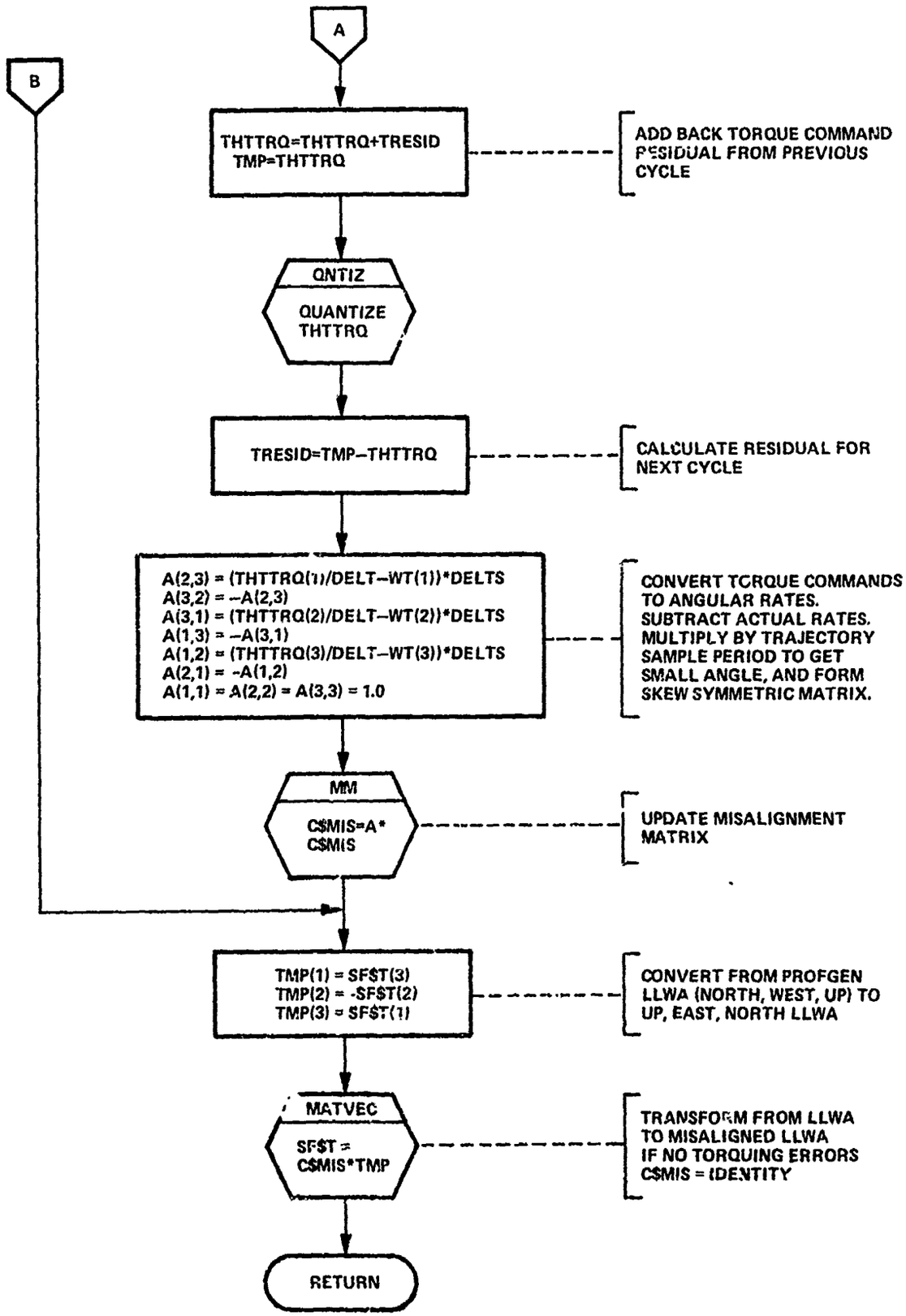
TMP - Temporary (scratch) vector.

Output Variables

C$MIS - 3x3 misalignment matrix. Referred to as $C_{2'}^{2m'}$ in above description.

SF$T - Specific force vector in LLWA or misaligned LLWA frame.


LLINTG
IOPNL=0
N
Y
B
GET AIRCRAFT VELOCITY IN NORTH AND EAST DIRECTIONS
$V_N = \cos\alpha_T * V_1^{2Q'} - \sin\alpha_T * V_2^{2Q'}$
$V_E = -\sin\alpha_T * V_1^{2Q'} - \cos\alpha_T * V_2^{2Q'}$
COMPUTE RADII OF CURVATURE RP, RM
WT(1) = WERT*SLAT$T
WTE = –VN/RM
WTN = VE/RP+WERT*CLAT$T
FIND ANGULAR RATES DUE TO EARTH ROTATION AND AIRCRAFT VELOCITY
WT(2) = WTE*CALF$T+WTN*SALF$T
WT(3) = –WTE*SALF$T+WTN*CALF$T
TRANSFORM RATES ABOUT EAST & NORTH TO LLWA FRAME
IPC(1) = 0
NO
YES
DOUT(1)
PRINT WT
MATVEC
WT=C$MIS *WT
TRANSFORM FROM LLWA TO MISALIGNED LLWA
A


A
B
THTTRQ=THTTRQ+TRESID
TMP=THTTRQ
ADD BACK TORQUE COMMAND RESIDUAL FROM PREVIOUS CYCLE
QNTIZ
QUANTIZE THTTRQ
TRESID=TMP–THTTRQ
CALCULATE RESIDUAL FOR NEXT CYCLE
A(2,3) = (THTTRQ(1)/DELT–WT(1))*DELTS
A(3,2) = –A(2,3)
A(3,1) = (THTTRQ(2)/DELT–WT(2))*DELTS
A(1,3) = –A(3,1)
A(1,2) = (THTTRQ(3)/DELT–WT(3))*DELTS
A(2,1) = -A(1,2)
A(1,1) = A(2,2) = A(3,3) = 1.0
CONVERT TORQUE COMMANDS TO ANGULAR RATES. SUBTRACT ACTUAL RATES. MULTIPLY BY TRAJECTORY SAMPLE PERIOD TO GET SMALL ANGLE, AND FORM SKEW SYMMETRIC MATRIX.
MM
C$MIS=A* C$MIS
UPDATE MISALIGNMENT MATRIX
TMP(1) = SF$T(3)
TMP(2) = -SF$T(2)
TMP(3) = SF$T(1)
CONVERT FROM PROFGEN LLWA (NORTH, WEST, UP) TO UP, EAST, NORTH LLWA
MATVEC
SF$T = C$MIS*TMP
TRANSFORM FROM LLWA TO MISALIGNED LLWA IF NO TORQUING ERRORS C$MIS = IDENTITY
RETURN

2.3.32 Function & Equations

The purpose of LLATT is to implement those calculations that would be performed in an airborne computer in constructing the $C_5^{2'}$ direction cosine matrix (transformation from body (5) frame to local-level (2') computational frame). The $C_5^{2'}$ matrix is constructed from the four gimbal angles generated by the LLATUD routine. The order in which the angles are used is given below:

Outer Roll (negative about Z)
Pitch (negative about Y)
Inner Roll (negative about Z)
Yaw (positive about X)

Two intermediate matrices are formed by using the subroutine ROTXYZ and then multiplying these two matrices to give $C_5^{2'}$. The equations are as follows:

$$C_5^{2'} = C_{3''}^{2'} C_5^{3''}$$

where:

$C_5^{3''}$ transforms through minus outer roll and minus pitch

$C_{3''}^{2'}$ transforms through minus inner roll and yaw.

$$C_5^{3''} = \begin{bmatrix} \text{cosXcosW} & -\text{cosXsinW} & \text{sinX} \\ \text{sinW} & \text{cosW} & 0 \\ -\text{sinXcosW} & \text{sinXsinW} & \text{cosX} \end{bmatrix}$$

$$C_{3''}^{2'} = \begin{bmatrix} \text{cosY} & -\text{sinY} & 0 \\ \text{cosZsinY} & \text{cosZcosY} & \text{sinZ} \\ -\text{sinZsinY} & -\text{sinZcosY} & \text{sinZ} \end{bmatrix}$$

where:

W - Outer roll
X - Pitch
Y - Inner roll
Z - Yaw

Input Variables

ETA$P - Outer roll, pitch, inner roll, yaw gimbal angles. Equivalenced to W,X,Y,Z respectively.

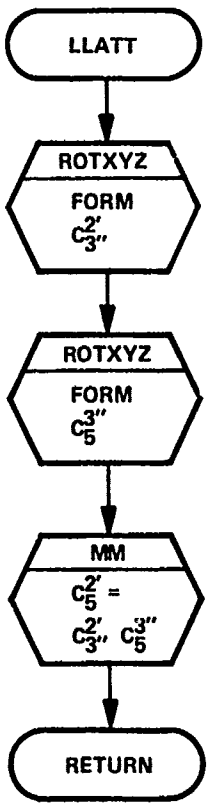
Internal Variables

A - 3x3 array which corresponds to $C_{3''}^{2'}$ above.

B - 3x3 array which corresponds to $C_{5}^{3''}$ above.

Output Variables

C2P$5 - Transformation matrix from body to local-level frames Corresponds to $C_{5}^{2'}$ above.


LLATT
ROTXYZ
FORM
$C_{3''}^{2'}$
ROTXYZ
FORM
$C_5^{3''}$
MM
$C_5^{2'}$ =
$C_{3''}^{2'}$, $C_5^{3''}$
RETURN

2.3.33 Function & Equations

The purpose of LLATUD is to generate the four gimbal angles that would be generated by a local-level IMU. Moving from the body frame (5) to the local-level frame (2'), the order of rotations is:

Outer Roll (negative about Z)
Pitch (negative about Y)
Inner Roll (negative about Z)
Yaw (positive about X)

where:

Positive roll is right wing down.
Positive pitch is nose up.
Positive yaw is counterclockwise about down.

Subroutine ROTXYZ is used to form the body to local-level transformation matrix (C2P$5). This is then multiplied by the misalignment matrix (C$MIS). C$MIS represents the difference between IMU orientation calculated from PROFGEN data and the IMU orientation caused by torquing commands from the navigation equations. If a perfect platform is being simulated, C$MIS is set to identity. Once C2P$5 has been formed, the gimbal angles can be extracted as follows:

$$\text{ETA\$P(1)} = \tan^{-1}\left[\frac{-\text{C2P\$5(1,2)}}{\text{C2P\$5(1,1)}}\right] = \text{outer roll}$$

$$\text{ETA\$P(2)} = \tan^{-1}\left[\frac{\text{C2P\$5(1,3)}}{\sqrt{\text{C2P\$5(2,3)}^2 + \text{C2P\$5(3,3)}^2}}\right] = \text{pitch}$$

$$\text{ETA\$P(3)} = \tan^{-1}\left[\frac{\text{C2P\$5(2,3)}}{\text{C2P\$5(3,3)}}\right] = \text{yaw}$$

After the above angles have been quantized or truncated to give the operator specified resolution, ETA$P(4) is set to ETA$P(3) since ETA$P(3) is actually the inner roll angle. ETA$P(3) is then set to zero.

Input Variables

ETA$T - Roll, pitch, and yaw from PROFGEN.

C$MIS - Misalignment matrix which transforms the C2P$5 matrix from the perfect local-level frame as generated by PROFGEN to the local-level frame generated as a result of torquing commands from the navigation equations.

AQUANT - Gimbal angle quantization term.

IPC(2) - Flag which controls operator selectable printout of ETA$P. If zero, no printout. Default is no printout.

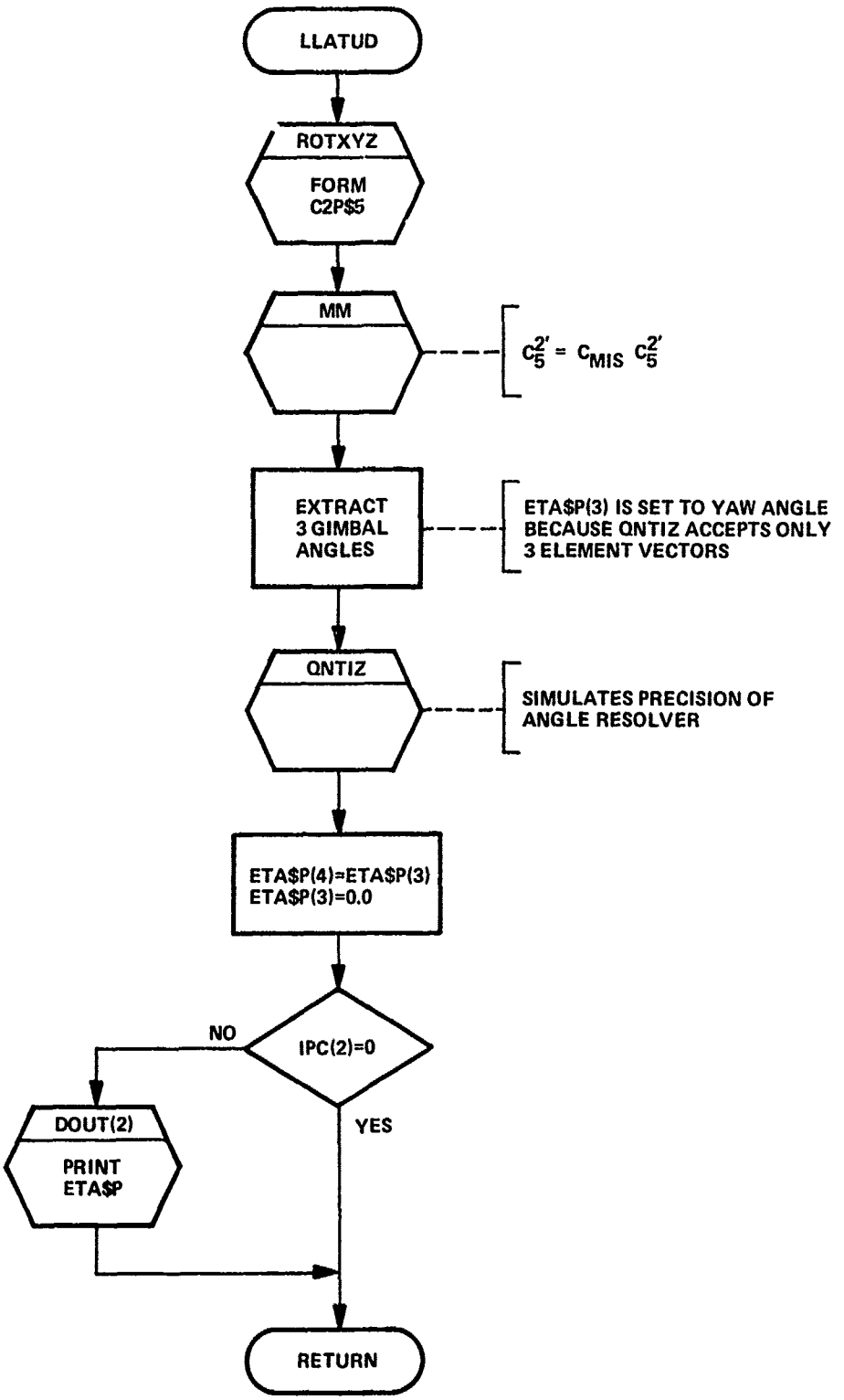
Internal Variables

TEMP - Temporary variable to hold the square root of C2P$5(2,3) squared plus C2P$5(3,3) squared.

C2P$5 - 3x3 rotation matrix from body to local-level wander azimuth (2') frame. Includes torquing misalignments if calculated.

Output Variables

ETA$P - Four local-level wander azimuth gimbal angles. Order is outer roll, pitch, inner roll, yaw. Inner roll is always zero.


LLATUD
ROTXYZ
FORM C2P$5
MM
$C_5^{2'} = C_{MIS}\ C_5^{2'}$
EXTRACT 3 GIMBAL ANGLES
ETA$P(3) IS SET TO YAW ANGLE BECAUSE QNTIZ ACCEPTS ONLY 3 ELEMENT VECTORS
QNTIZ
SIMULATES PRECISION OF ANGLE RESOLVER
ETA$P(4)=ETA$P(3)
ETA$P(3)=0.0
IPC(2)=0
NO
YES
DOUT(2)
PRINT ETA$P
RETURN

2.3.34 Function & Equations

The purpose of LLINIT is to perform the initialization needed by the local level IMU model and the navigation equations for local level input. For the most part this initialization is performed by a series of calls to the appropriate subroutines (see flow charts). The only calculations performed in LLINIT are as follows:

ICMCYL = ITRNAV*ITRATT

where:

ICMCYL = Number of trajectory input cycles which must be integrated between each navigation cycle.

ITRNAV = Number of attitude calculation cycles per navigation cycle.

ITRATT = Number trajectory inputs per attitude cycle.

The second calculation establishes the first past value of the specific forces which are input from PROFGEN. The specific forces are transformed from the PROFGEN frame (North, West, Up) to the simulator local level frame (Up, East North).

$$\begin{bmatrix} SF\$TP(1) \\ SF\$TP(2) \\ SF\$TP(3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} SF\$T(1) \\ SF\$T(2) \\ SF\$T(3) \end{bmatrix}$$

where:

SF$TP = Past value of specific force in local-level frame (U.E.N.).

SF$T = Present value of specific force as input from PROFGEN (N.W.U.).

Input Variables

ITRNAV
ITRATT
SF$T
} See above for definition

IOPNLP - Control flag to establish whether or not torquing

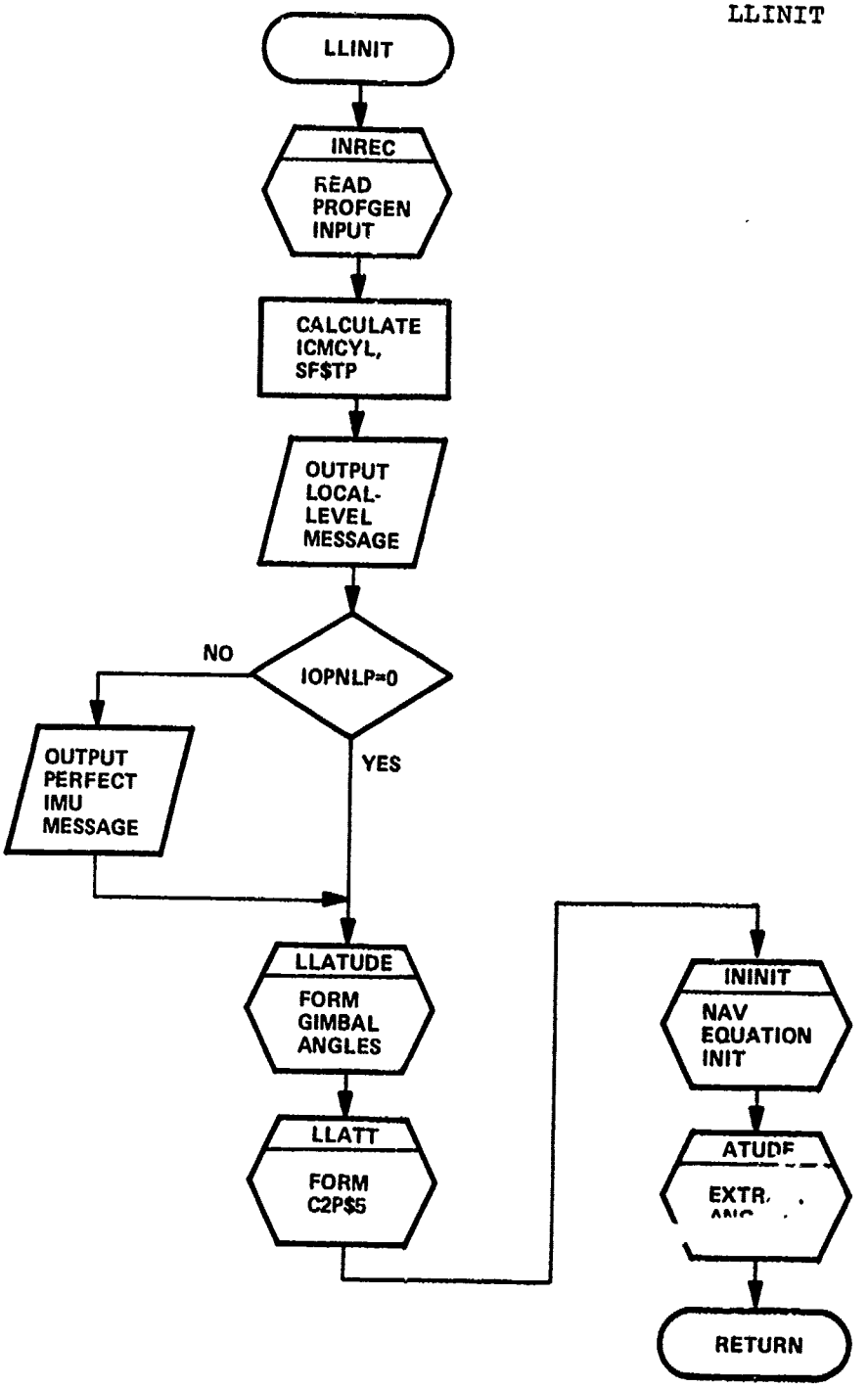
misalignment within the local-level IMU model
is to be calculated. 0 = calculate misalignment;
1 = no misalignment.

Internal Variables

None

Output Variables

ICMCYL, SF$TP } See above for definition


LLINIT
INREC
READ PROFGEN INPUT
CALCULATE ICMCYL, SF$TP
OUTPUT LOCAL-LEVEL MESSAGE
IOPNLP=0
NO
YES
OUTPUT PERFECT IMU MESSAGE
LLATUDE
FORM GIMBAL ANGLES
LLATT
FORM C2P$5
ININIT
NAV EQUATION INIT
ATUDE
EXTR
RETURN

SSINIT

2.3.35 Function & Equations

The purpose of SSINIT is to perform initialization when space stable IMU is selected (IMUTYP = 2). The sequence of initializations performed is as follows (note that much initialization is 'performed' in the BLOCK DATA routine).

A) Compute C_{10}^{0} rotation based on rotations, in the Euler angle sense, of SSX0 about X, followed by SSY0 about Y, followed finally by SSZ0 about Z.

B) Compute C_{0}^{10} by transposing C_{10}^{0}

C) Initialize C_{0}^{0P} to identity

D) INREC is called. This initializes all variables to all the rates (DELT, DELTS, etc):

E) ICMCYL is initialized so that the CMINTG integration cycle is the navigation cycle.

F) C_{2P}^{0P} is initialized

G) SSINTG is called to rotate the TIME0 specific force into the space stable frame, which is subsequently stored in SF$TP.

H) An initialization message is printed

I) SSATUD is called to compute the initial gimbal angles.

J) SSATT is called to compute the initial C2P$5

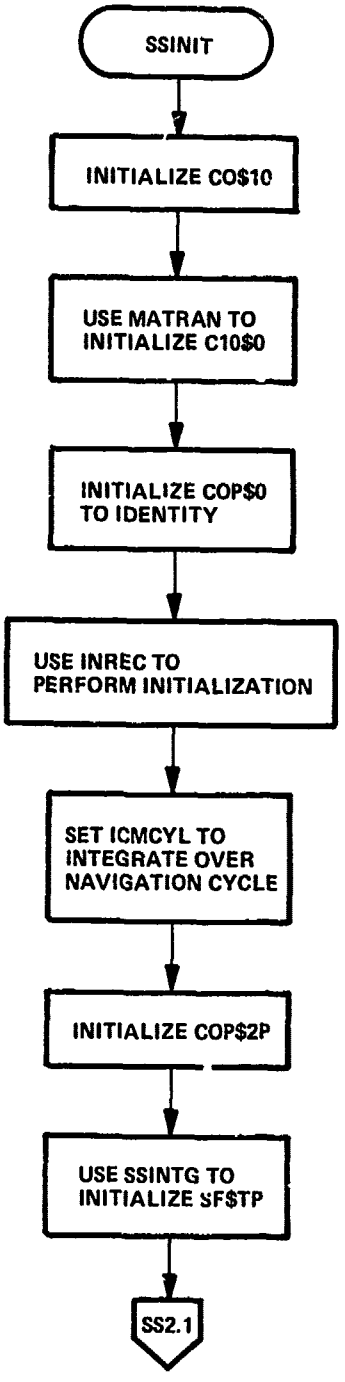
K) ININIT is called to initialize the navigation equations.

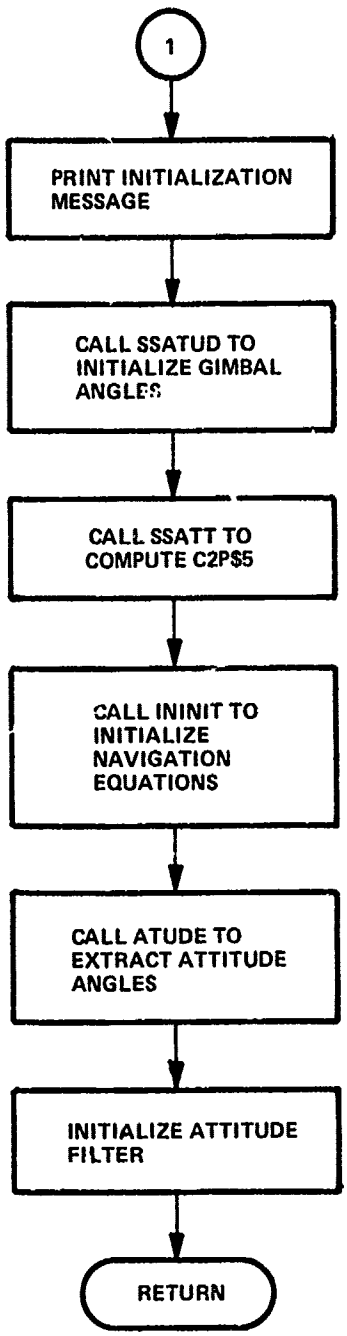
L) ATUDE is called to extract the gimbal angles

M) The attitude filter is initialized, if necessary.

Inputs, Outputs and Internal Variables

All variables not previously initialized are initialized.


SSINIT
INITIALIZE CO$10
USE MATRAN TO INITIALIZE C10$0
INITIALIZE COP$0 TO IDENTITY
USE INREC TO PERFORM INITIALIZATION
SET ICMCYL TO INTEGRATE OVER NAVIGATION CYCLE
INITIALIZE COP$2P
USE SSINTG TO INITIALIZE SF$TP
SS2.1


1
PRINT INITIALIZATION MESSAGE
CALL SSATUD TO INITIALIZE GIMBAL ANGLES
CALL SSATT TO COMPUTE C2P$5
CALL ININIT TO INITIALIZE NAVIGATION EQUATIONS
CALL ATUDE TO EXTRACT ATTITUDE ANGLES
INITIALIZE ATTITUDE FILTER
RETURN

SSATUD

2.3.36 Function & Equations

The purpose of SSATUD is to compute the gimbal angles that would be generated by a perfect space stable IMU. The IMU is assumed to be a four gimbal platform, with the rotations about Z, X, Y and Z, in that order, going from the body in towards the IMU. The Y rotation is assumed to be zero, so that the rotation is a Z X Z rotation.

The first step is to form the C2P$5 matrix by using the ROTXYZ routine and the roll, pitch and yaw angles from PROFGEN.

A Z(α_1) X (α_3) Z (α_4) rotation can be written as follows (where S_1 = sine (α_1), C_1 = cosine (α_1), S_3 = sine (α_3), C_3 = cosine (α_4) C_4 = cosine (α_4):

$$A = \begin{bmatrix} C1C4 - S1C3S4 & C1S4 + S1C3C4 & S1S3 \\ -S1C4 - C1C3S4 & -S1S4 + C1C3C4 & C1S3 \\ S3S4 & -S3C4 & C3 \end{bmatrix}$$

The angles (α_1,α_3, and α_4) are there follows:

1. Compute S_3 as $\sqrt{(A\ (3,1))^2 + (A\ (3\ 2))^2}$

2. If S_3 is not $\neq 0$, compute

$$\alpha_1 = \tan^{-1}\left(\frac{A(1,3)}{A(2,3)}\right)$$

$$\alpha_3 = \tan^{-1}\left(\frac{S_3}{A(3,3)}\right)$$

$$\alpha_4 = \tan^{-1}\left(\frac{A(3,1)}{-A(3,2)}\right)$$

We wish to keep α_3 between 0° and 180° (to avoid a situation known as "gimbal flip"). Therefore if either α_1 or α_4 has changed by more than $\Pi/2$, do the following

$$\alpha_4 = \alpha_4 + \pi$$

$$\alpha_3 = -\alpha_3$$

$$\alpha_1 = \alpha_1 + \pi$$

To allow for the way the outer gimbal is usually mounted subtract $\pi/4$ from α.

Now quantize the attitude angles in units of AQUANT, using the QNTIZ subroutine.

Inputs, Outputs & Internal Variables

Inputs:

ETA$T - Roll, pitch and yaw from PROFGEN

C10$2P - tranformation from computational to space stable frame (computed each DELTS by SSINTG)

AQUANT - angle quantization units

Input and Output:

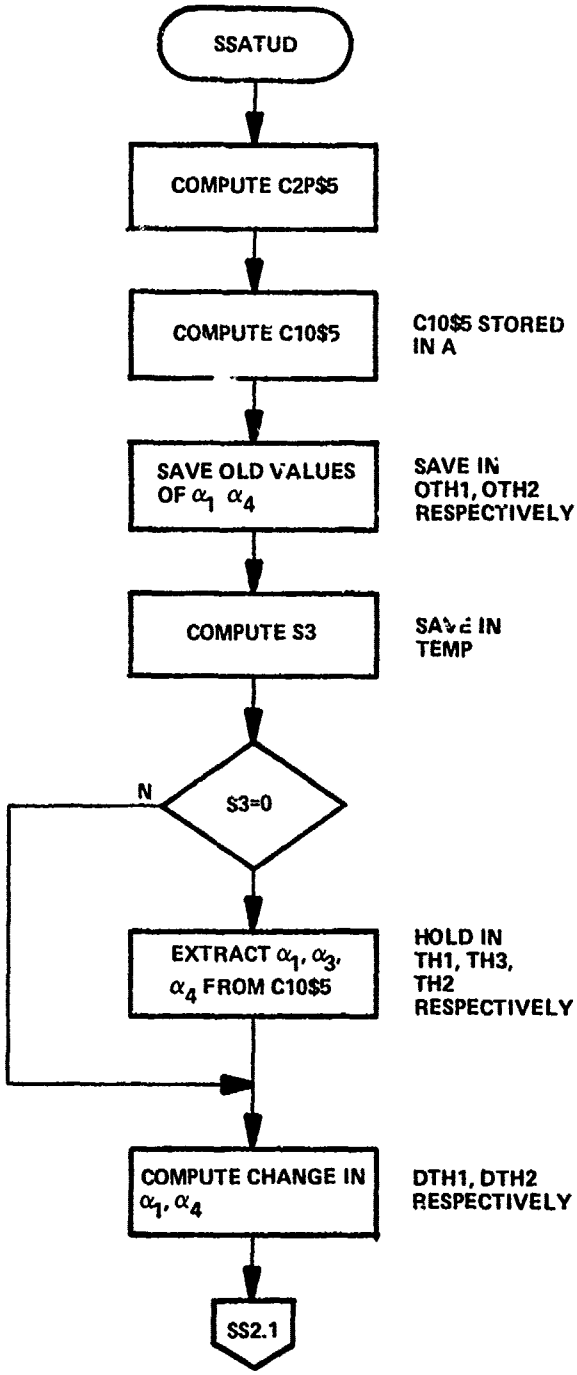
ETA$P - Old, on input, values of SS gimbal angles. Output as new values.

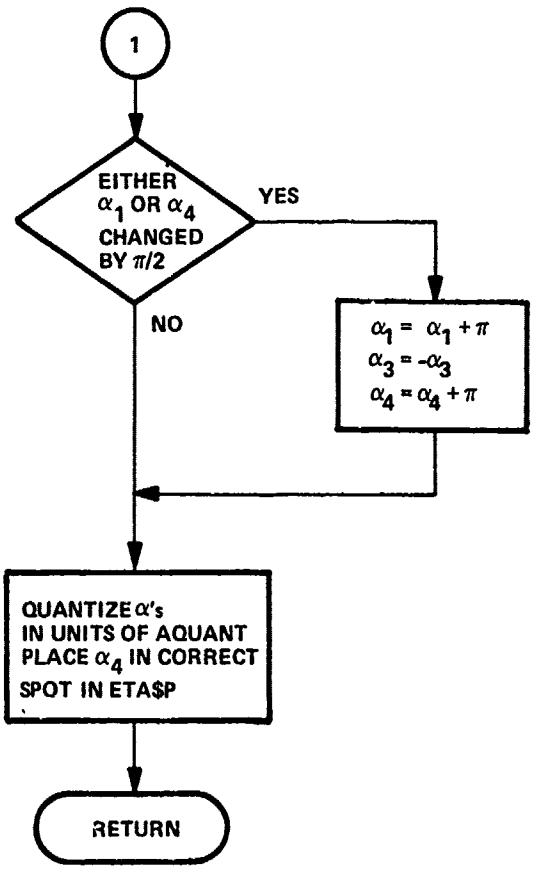
Internal Variables:

TH1, TH2, TH3 - EQUIVALENCEd to ETA$P(1) etc.

A - holds C10$5

OTH2, OTH1, DTH1, DTH2 - internal scratch variables


SSATUD
COMPUTE C2P$5
COMPUTE C10$5
C10$5 STORED IN A
SAVE OLD VALUES OF α_1 α_4
SAVE IN OTH1, OTH2 RESPECTIVELY
COMPUTE S3
SAVE IN TEMP
S3=0
N
EXTRACT α_1, α_3, α_4 FROM C10$5
HOLD IN TH1, TH3, TH2 RESPECTIVELY
COMPUTE CHANGE IN α_1, α_4
DTH1, DTH2 RESPECTIVELY
SS2.1


1
EITHER α_1 OR α_4 CHANGED BY $\pi/2$
YES
NO
$\alpha_1 = \alpha_1 + \pi$
$\alpha_3 = -\alpha_3$
$\alpha_4 = \alpha_4 + \pi$
QUANTIZE α's IN UNITS OF AQUANT PLACE α_4 IN CORRECT SPOT IN ETA$P
RETURN

SSTFRM

2.3.37 Function and Equations

The purpose of this routine is to implement that portion of the flight code (of a space stable system only) which would transform the delta -V's from the space stable to the computational frame. This routine is always called just prior to INRNAV. The basic equation is:

$$\underline{f}^{2P} = C_{0P}^{2P} \; C_{0}^{0P} \; C_{10}^{0} \; \underline{f}^{10}$$

C_{10}^{0} is set up at initialization and, with the assumption of a perfect space-stable IMU, remains constant throughout time. C_{0}^{0P} is just the rotation matrix resulting from the rotation of the earth since TIME0; it is computed to the middle of the current navigation cycle. C_{0P}^{2P} is computed as follows:

i) If IPC(29) = 1 X0P$2P contains C_{2P}^{0P} at the middle of the current navigation cycle; the transpose of X0P$2P is used

ii) Otherwise (that is IPC(29) = 0), C_{0P}^{2P} is computed as follows:

$$C_{0P}^{2P} = \left[C_{2P}^{0P} + \frac{1}{2} \; C_{2P}^{0P} \times \underline{\rho}_{0P,2P}^{2P} \Delta t \right]^{T}$$

where C_{2P}^{0P} is the value (as computed by INRNAV and stored in C0PS2P of the beginning of the current navigation cycle, and $\underline{\rho}_{0P,2P}^{2P}$ (as computed by INRNAV and stored, in skew-symetric matrix form as C$RHO) is the angular rate.

Inputs, Outputs & Internal Variables

Inputs:

DV$P - delta V

CO$10 - transformation from space stable to 'TIME0 equatorial Up-East-North'

WERT - earth rate

TIME0 - initial TIME

TIME - current TIME

DELT - navigation cycle

X0P$2P- middle of cycle C_{2P}^{0P}

CSRHO - skew symetric form of RHO*DELT

C0P$2P- beginning of cycle C_{2P}^{0P}

Outputs:

DV$P - delta V's, now in computational frame

C0P$0 - DCM from 0 to 0P frame

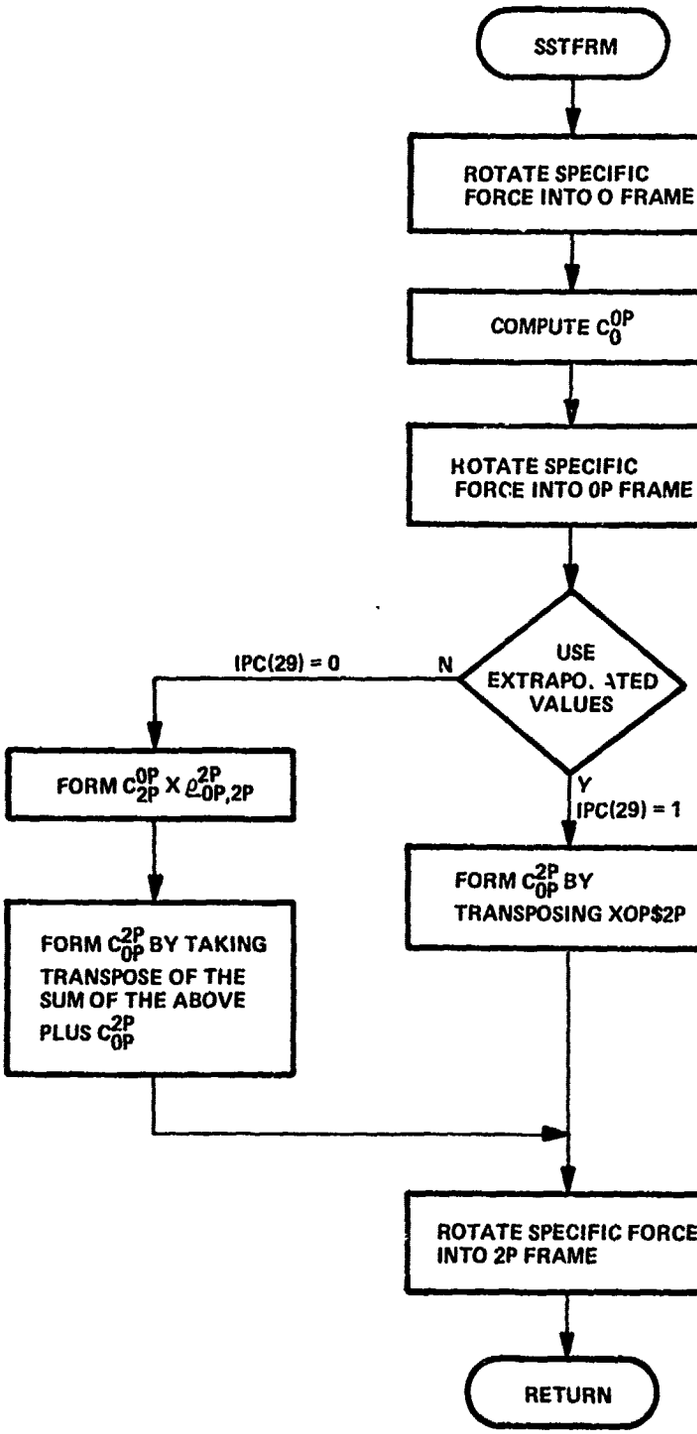
Internal Variables:

WET - rotation of earth since TIME0

A - holds 1/2 $C_{2P}^{0P} \times \underline{\rho}^{2P}_{0P,\ 2P}$

B - holds C_{0P}^{2P}

TEMP, TEMP2 - scratch vectors


SSTFRM
ROTATE SPECIFIC FORCE INTO O FRAME
STORED IN TEMP
COMPUTE C_0^{0P}
ROTATE SPECIFIC FORCE INTO 0P FRAME
USE EXTRAPO. ATED VALUES
N
IPC(29) = 0
Y
IPC(29) = 1
FORM $C_{2P}^{0P} \times \varrho_{0P,2P}^{2P}$
FORM C_{0P}^{2P} BY TAKING TRANSPOSE OF THE SUM OF THE ABOVE PLUS C_{0P}^{2P}
FORM C_{0P}^{2P} BY TRANSPOSING XOP$2P
ROTATE SPECIFIC FORCE INTO 2P FRAME
RETURN

SSINTG

2.3.38 Function & Equations

The purpose of this routine is to transform the specific forces from the PROFGEN platform frame to the space stable frame; as such it is part of the TNT simulator. The equation used is as follows:

$$\underline{f}^{10} = C_{2P}^{10} \; C_{2QP}^{2P} \; \underline{f}^{2QP}$$

(2QP is PROFGENS platform frame)

the routine also forms C_{2P}^{10} which is used by SSATUD. The C_{2QP}^{2P} tranformation consists only of

i) interchanging the first and third components and

ii) negating the second component.

C_{2P}^{10} is computed as

$$C_{2P}^{10} = C_{0}^{10} \; C_{2P}^{0}$$

C_{0}^{10} is available as an input; it represents the orientation of the platform in inertial space, and as such, does not change with time.

C_{2P}^{0} is the transformation from platform to "equatorial Up-East-North at TIMEØ". It is a rotation of alpha about up (note that PROFGENs alpha has the opposite sign convention), followed by a rotation (in the Euler angle sense) of latitude about east, followed by a minus rotation of "longitude change" about north. "Longitude change" is computed as

WERT * (TIME - TIMEØ) + LONG$T - LONGØ

where

WERT is earth angular velocity in radians/sec

TIME is current (end of cycle) time

TIMEØ is initial time

LONGØ is longitude at TIMEØ

LONG$T is current (end of cycle) longitude as computed by PROFGEN

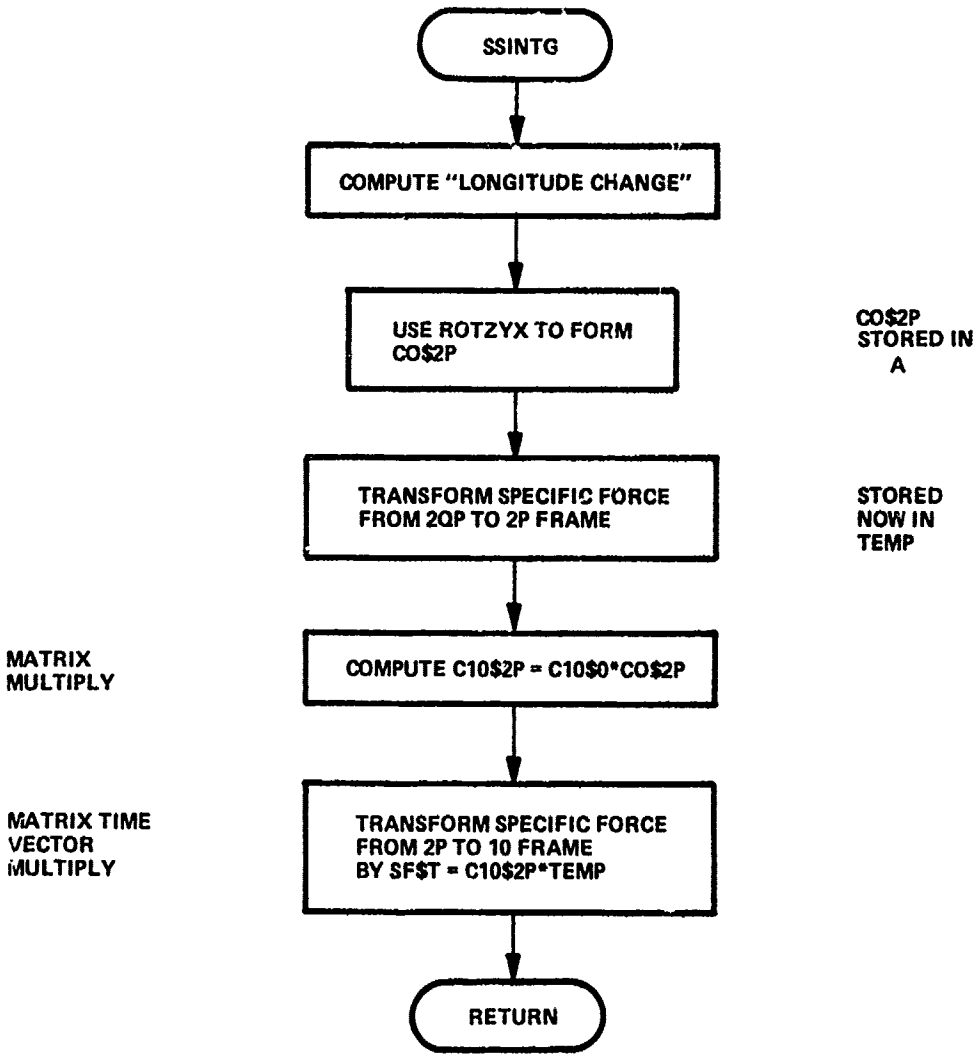
Inputs, Outputs & Internal Variables

Inputs:

WERT	
TIME	
TIMEØ	
LONGØ	
LONG$T	
C10$0	
SLAT$T	sine and cosine of latitude as computed by PROFGEN
CLAT$T	
SF$T	PROFGENS specific forces in 2QP frame

Ouputs:

C10$2P	
SF$T	Specific force in space stable frame


SSINTG
COMPUTE "LONGITUDE CHANGE"
USE ROTZYX TO FORM CO$2P
CO$2P STORED IN A
TRANSFORM SPECIFIC FORCE FROM 2QP TO 2P FRAME
STORED NOW IN TEMP
MATRIX MULTIPLY
COMPUTE C10$2P = C10$0*CO$2P
MATRIX TIME VECTOR MULTIPLY
TRANSFORM SPECIFIC FORCE FROM 2P TO 10 FRAME BY SF$T = C10$2P*TEMP
RETURN

SSATT

2.3.39 Function & Equations

The purpose of this routine is to implement those calculations that would be performed by an airborne computer in constructing the C2P$5 DCM (transformation from body to computational frame). The transformation C10$5 (from body to space stable frame) can be constructed by a rotation of $(\alpha_1 + \pi/2)$ about Z, followed by (in the Euler Angle sense) a rotation of α_3 about X, followed in turn by a rotation of α_4 about Z. The α_1, α_3 and α_4 are the gimbal angles read in from a space stable IMU.

A $Z(\alpha_1)$ X (α_3) Z (α_4) can be written as follows (where S_1 = sine (α_1), C_1 = cosine (α_1), S_3 = sine (α_3), C_3 = cosine (α_3), S_4 = sine (α_4), C_4 = cosine (α_4).

$$\begin{bmatrix} C1C4 - S1C3S4 & C1S4 + S1C3C4 & S1S3 \\ -S1C4 - C1C3S4 & -S1S4 + C1C3C4 & C1S3 \\ S3S4 & -S3C4 & C3 \end{bmatrix}$$

Once we have C10$5 we can use COP$2P, C0$10 and COP$0 (which are available as inputs) as follows:

$$C2P\$5 = [C0P\$2P]^T * [C0P\$0] * [C0\$\ 10] * [C10\$5]$$

Inputs, Outputs & Interval Variables

Inputs:

C0P$2P - DCM from 2P to 0P
C0P$0 - DCM from 0 to 0P
C0$10 - DCM from 10 to 0
ETA$P - gimbal angles (generated by SSATUD)

Outputs

C2P $5 - DCM from 5 to 2P

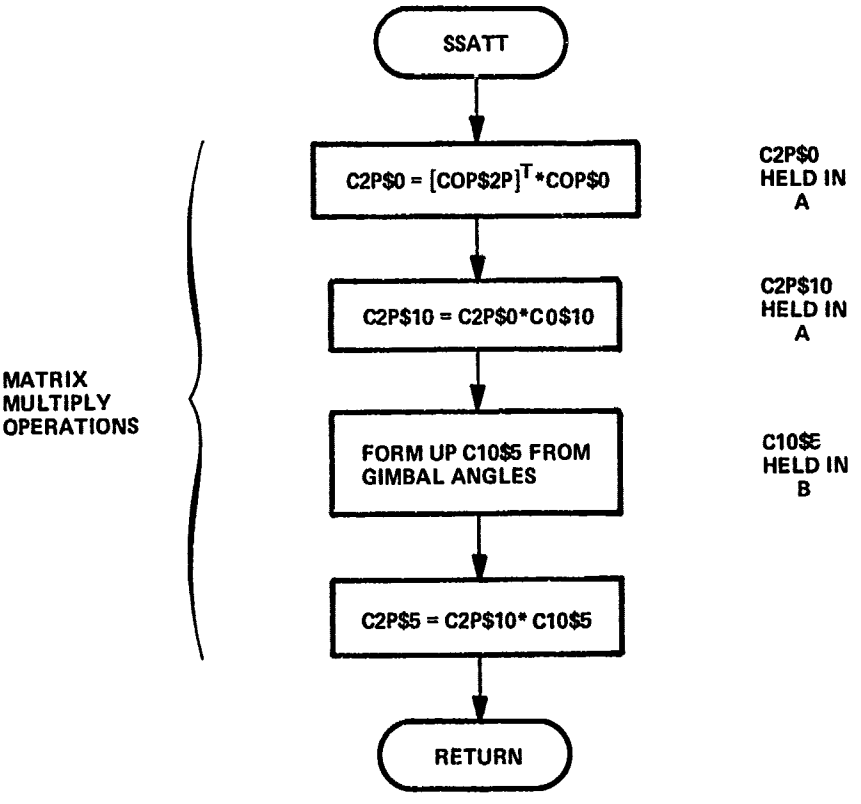
Internal Variables

A - holds C2P $0, then C2P $10

B - holds C10 $5

S1, C1, S3
C3, S4, C4
S1S4, C1C3
S3S4, C1C4

Scratch variables used in creating C10$5


SSATT
C2P$0 = [COP$2P]T*COP$0
C2P$0 HELD IN A
C2P$10 = C2P$0*C0$10
C2P$10 HELD IN A
FORM UP C10$5 FROM GIMBAL ANGLES
C10$E HELD IN B
C2P$5 = C2P$10* C10$5
RETURN
MATRIX MULTIPLY OPERATIONS

2.3.40 Function And Equations

The purpose of this subroutine is perform rounding (optional) and truncation of floating point numbers; this is necessary to simulate an airborne computer whose precision is less than that of the 'host' computer (the host is the computer on which the simulation is being run, a CYBER 74). For a detailed description of the algorithms, refer to the comments in the listing.

Inputs, Outputs & Internal Variables

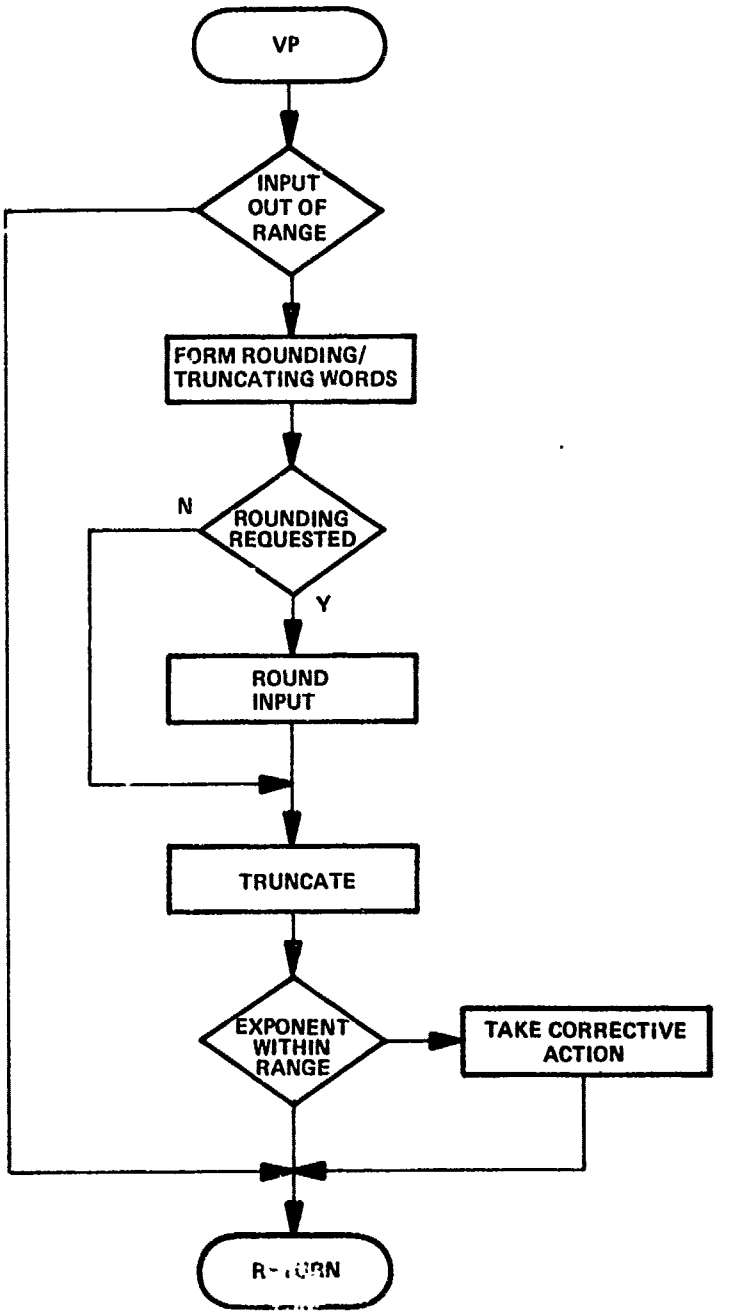
Input argument - remember, this is an assembly language subroutine

FRACTL - length of fraction being simulated

ROUND - odd implies rounding, even implies truncation only

Output result

AS DEFINED BY CDC HARDWARE



VPINIT

2.3.41 Function & Equations

The purpose of this routine is to initialize the precision varying routine with information concerning the length of the exponent of the floating point numbers to be simulated. For a more complete description, refer to the comments in the routine listing in Volume IV.

Inputs, Outputs & Internal Variables

Input - EXPOL - length of exponent

VSIN

2.3.42 Function & Equations

The purose of VSIN is to compute the sine of a single argument. VSIN implements the sine routine that would be implemented in an airborne computer. The algorithm to be used is selected by specifying SINPTH to be 1, 2, 3 or 4; these algorithms are designed to produce, respectively, 24, 32, 40 or 48 bits of accuracy in the result. VSIN sets an indicator, and then calls VSINCO (a subroutine that it shares with VCOS) to perform the actual computations. For a detailed description and flowchart, see VSINCO documentation, section 2.3.44.

Inputs, Outputs & Internal Variables

ARG - input argument
VSIN - result

2.3.43 Function & Equations

The purpose of VCOS is to compute the cosine of a single argument. VCOS implements the cosine routine that would be implemented in an airborne computer. The algorithm to be used is selected by specifying COSPTH to be 1, 2, 3, or 4; these algorithms are designed to produce respectively, 24, 32, 40 or 48 bits of accuracy in the result. VCOS sets an indicator, and then calls VSINCO (a subroutine that it shares with VSIN) to perform the actual computations. For a detailed description and flowchart, see VSINCO documentation, section 2.3.44.

Inputs, Outputs & Internal Variables

ARG - input argument
VCOS - result value

VSINCO

2.3.44 Function & Equations

The purpose of this routine is to compute the sine or cosine of an input argument. Sine vs. cosine is selected by an input argument, and the algorithm is likewise selected by an input argument. This routine is used by both VSIN and VCOS; VSINCO implements the calculations that would be performed by the airborne computer.

The algorithm input argument will have the value 1, 2, 3 or 4 which will select an algorithm that is accurate to, respectively, 24, 32, 40 or 48 bits.

It appears that most SIN/COS routines are implemented either as polynomials, or as ratios of polynomials. These polyonomials are evaluated over a reduced range as follows:

Assume X is the argument, as expressed in radians.

$z = \frac{4}{\pi}$ abs (X) $\qquad\qquad$ q = gii (z)
r = z-q

If VSIN was requested, and X<0 add 4 to q
If VCOS was requested add 2 to q

Let q_0 = q mod 8
Let q_1 = gii (q_0 /4)
Let q_2 = q_0 mod 4

Then the result (for either VSIN or VCOS) is as follows:

for q_2 = 0 $(1)^{q_1}$ sin $(\frac{\pi}{4}$ r)
for q_2 = 1 $(-1)^{q_1}$ cos $(\frac{\pi}{4}$ (1-r)
for q_2 = 2 $(-1)^{q_1}$ cos $(\frac{\pi}{4}$ 4)
for q_2 = 3 $(-1)^{q_1}$ sin $(\frac{\pi}{4}$ (1-r)

Simplifying further

$$r_1 = \begin{cases} r & \text{for } q_2 \bmod 2 = 0 \\ 1-r & \text{for } q_2 \bmod 2 = 1 \end{cases}$$

$$q_4 = \begin{cases} 0 & \text{for } q_2 = 0,3 \\ 1 & \text{for } q_2 = 1,2 \end{cases}$$

Then the result is

for $q_4 = 0$ $\quad (-1)^{q_1} \sin\left(\frac{\Pi}{4} r_1\right)$

for $q_4 = 1$ $\quad (-1)^{q_1} \cos\left(\frac{\Pi}{4} r_1\right)$

Various expansions (both polynomial and ratio of polynomials are available for sine $(\frac{\Pi}{4}$ r1) $0 \leq$ r1 <1 (likewise for cos $(\frac{\Pi}{4}$ r1). Several of these are listed in the reference (" Computer Approximations", Hart et al QA297,C738).

Assuming the relative execution times as follows:

ADD	1
MUL	3
DIV	10

the expansions for sin $(\frac{\Pi}{4} r_1)$ compare as follows:

Index in Reference	Precision (in decimal digits) (relative)	Relative Execution
3040	8.49	12
3041	11.34	16
3042	14.35	20
3043	17.48	24
3044	20.73	28
3045	24.07	32
3060	8.63	22
3061	11.36	26
3062	14.62	30
3063	17.59	34
3064	21.13	38
3065	24.29	42

We desire routines that will produce answers accurate to 24, 32, 40 and 48 binary digits (approximately 7.23, 9.61, 11.99, and 14.37).

With the same assumptions, the cosine expansion compare as follows:

Index in Reference	Precision (in decimal digits) (relative)	Relative Execution
3820	7.49	12
3821	10.25	16
3822	13.18	20
3823	16.25	24
3824	19.45	28
3825	22.74	32
3840	7.54	22
3841	10.10	26
3842	13.34	30
3843	16.18	34
3844	19.71	38
3845	22.78	42

Therefore define paths as follows:

PATH	COS Expansion	SIN Expansion	Accuracy in Decimal Digits (relative)
1	3820	3040	7.49
2	3821	3041	10.25
3	3822	3042	13.18
4	3823	3043	16.25

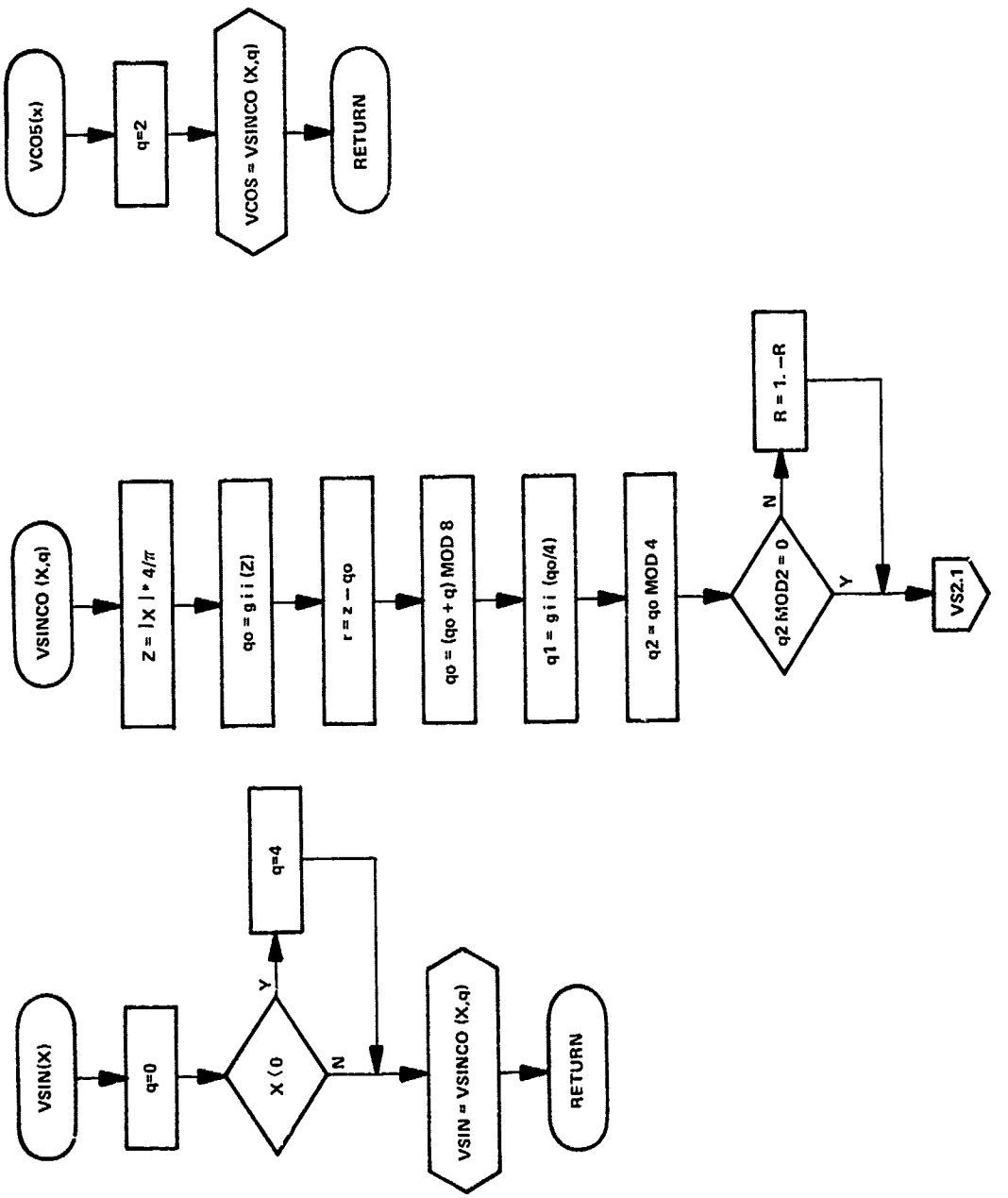
Inputs, Outputs, and Internal Variables

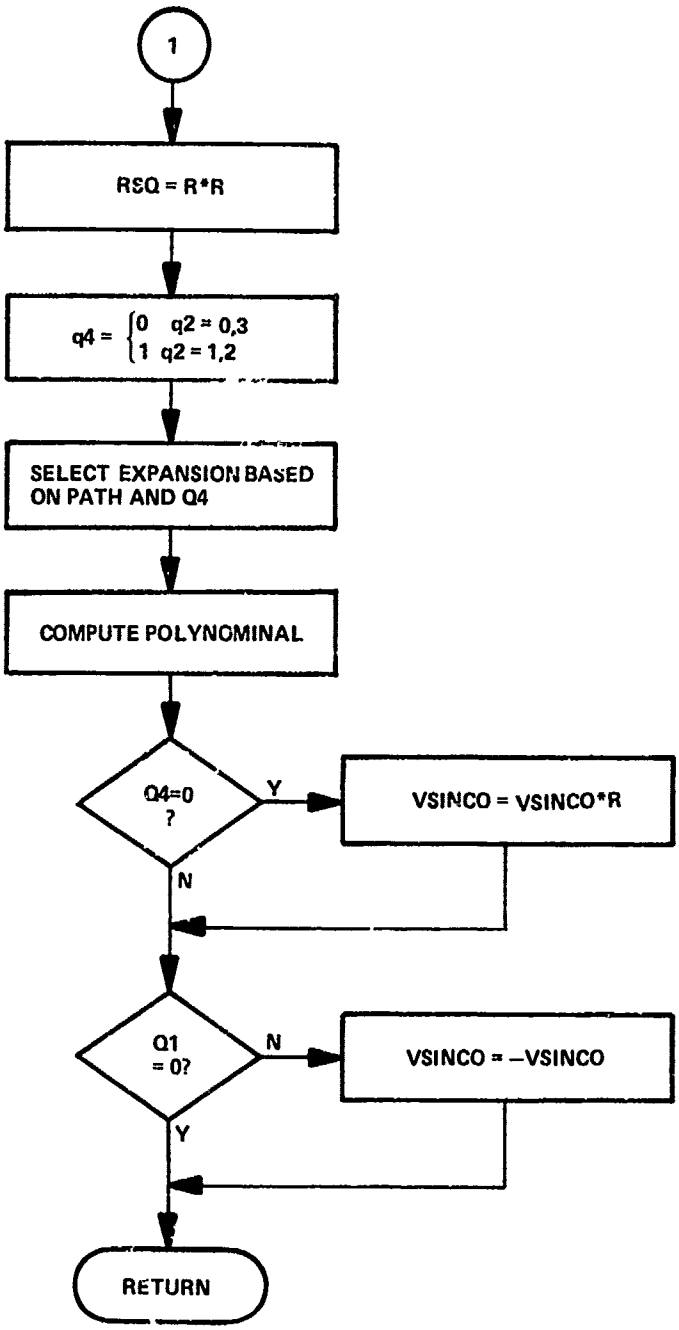
X - input argument

Q - 0 implies we want sin abs (x)
2 implies we want cos abs (x)
4 implies we want -sin abs (x)

IPATH - algorithm to use (1, 2, 3 or 4)

VSIN/VCOS FLOWCHART

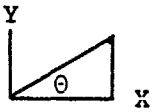



1
RSQ = R*R
q4 = 0 q2 = 0,3
1 q2 = 1,2
SELECT EXPANSION BASED ON PATH AND Q4
COMPUTE POLYNOMINAL
Q4=0 ?
Y
VSINCO = VSINCO*R
N
Q1 = 0?
N
VSINCO = –VSINCO
Y
RETURN

2.3.45 Function & Equations

The purpose of VATAN2 is to compute the arc tangent of two arguments, that is, $\tan^{-1}\left(\frac{X}{Y}\right)$. VATAN2 implements the arctangent routine that would be implemented in an airborne computer. The algorithm to be used is selected by specifying ATNPTH to be 1, 2, 3, or 4; these algorithms are designed to produce, respectively, at least 24, 32, 40 or 48 bits of accuracy in the result. The only error condition is if both Y and X are zero; in such a case the standard Fortran arctangent error routine is invoked.

VATAN2 takes two arguments Y and X, which can be graphically represented



where Θ is the arc tangent and $-\Pi < \Theta \leq \Pi$

if X > 0 VATAN 2 (Y, X) = sgn*(Y) atan (| Y |, | X |)

where atan (| Y |, | X |) = $\tan^{-1}$ (| Y |/| X |) and the principal value is between 0 and $\frac{\Pi}{2}$

if X < 0 VATAN2 (Y, X) = sgn*(Y) (Π - atan (| Y |, | X |))

if $|X| \geq |Y|$ atan (| Y |, | X |) = $\tan^{-1}$ (| Y |/ | X |)

if $|Y| \leq |X|$ atan (| Y |, | X |) = $\frac{\Pi}{2}$ -$\tan^{-1}$ (| X |/| Y |)

* sgn is 1 or -1, |X| is the absolute value of X

The argument of $\tan^{-1}$ is between 0 and 1, Various further range reductions are possible; one of them does not require any table searches. The algorithm is as follows:

if arg < $\tan \frac{\pi}{12} = 2 - \sqrt{3}$

then result is between 0 and $\frac{\pi}{12}$

if arg > $\tan \frac{\pi}{12}$

then $\tan^{-1}(\text{arg}) = \frac{\pi}{6} + \tan^{-1}\left(\frac{[\sqrt{3} - 1]\,\text{arg} - 1 + \text{arg}}{\text{arg} + \sqrt{3}}\right)$

where the argument satisfies

$$-\tan \frac{\pi}{12} < \tan < \tan \frac{\pi}{12}$$

The reference ("Computer Approximations", Hart, et.al., QA297.:738) lists several expansions for this range.

Assuming relative execution times as follows:

ADD	1
MUL	3
DIV	10

The expansions are as follows:

Index in Reference	Precision in Decimal Digits (relative)	Execution Time
4940	7.69	8
4941	9.54	12
4942	11.38	16
4943	13.20	20
4944	15.01	24
4945	16.82	28
4946	18.63	32
5050	6.62	18
5051	8.98	22
5052	11.34	26
5053	13.70	30
5054	16.06	34
5055	18.42	38
5056	20.78	42
5057	23.14	46
5058	25.51	50

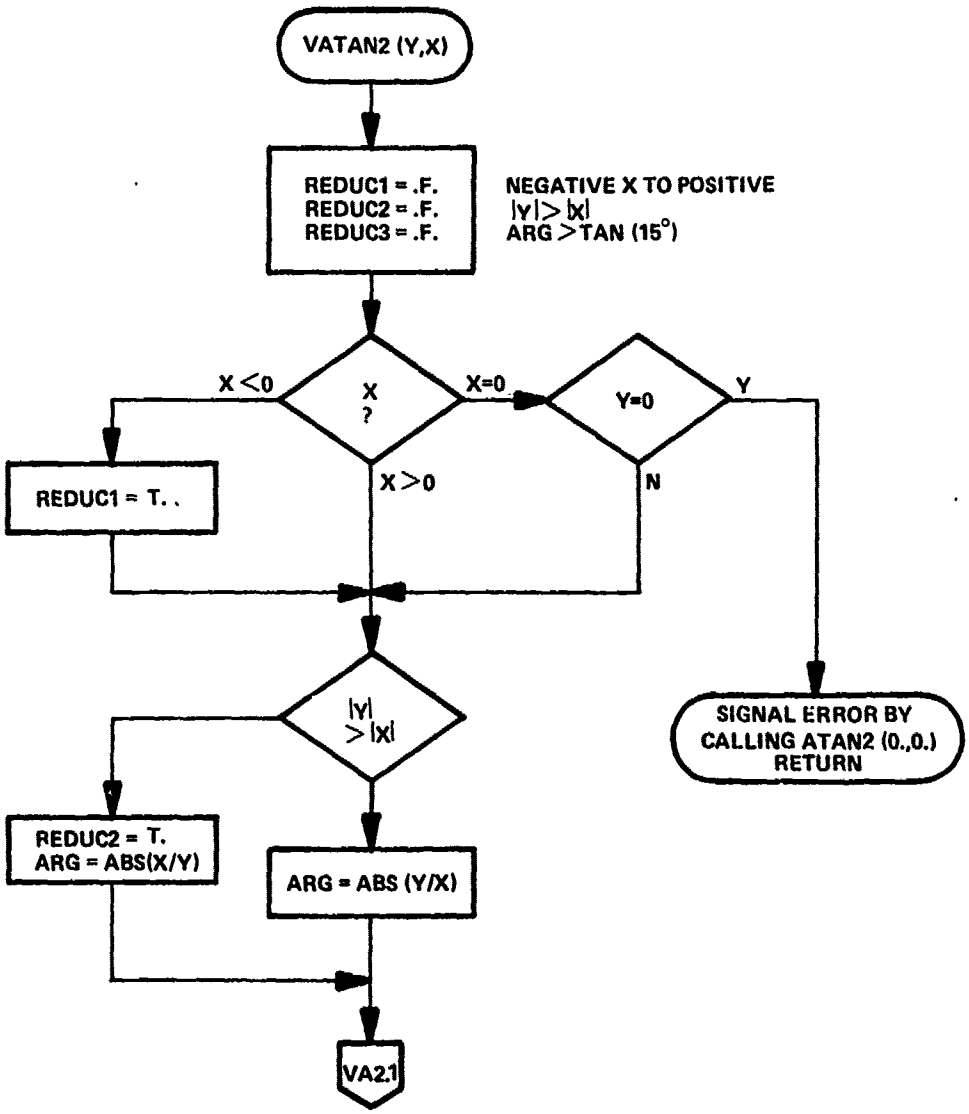
We desire routines that will produce answers accurate to 24, 32, 40 and 48 binary digits (approximately 7.23, 9.61, 11.99 and 14.37 decimal digits).

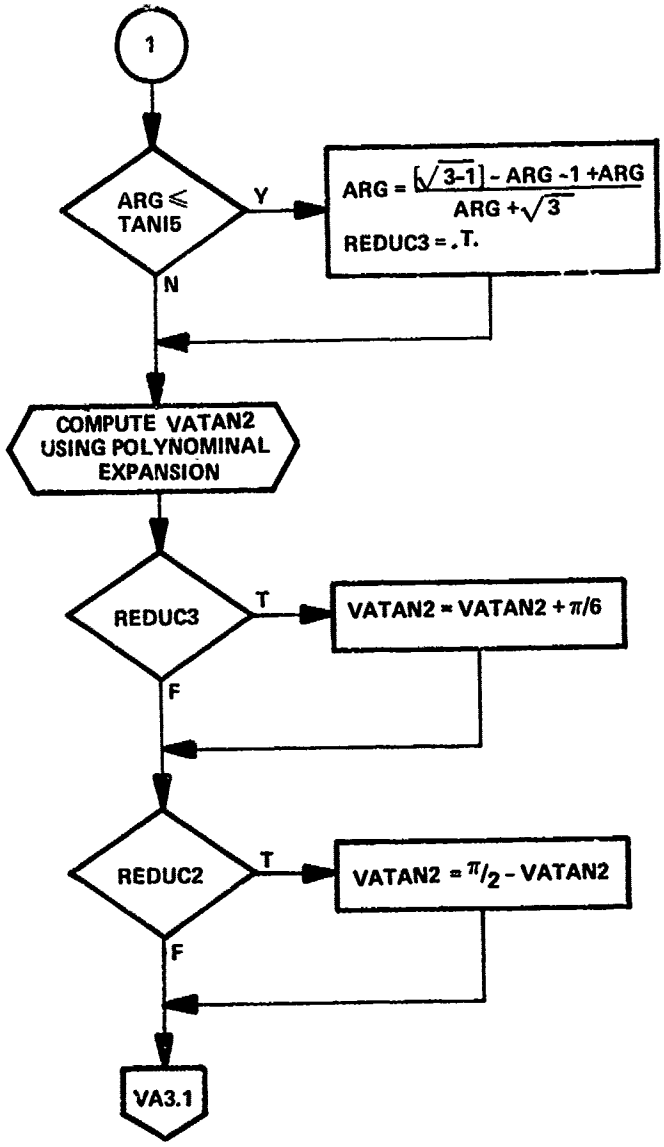
Therefore we will define the following paths:

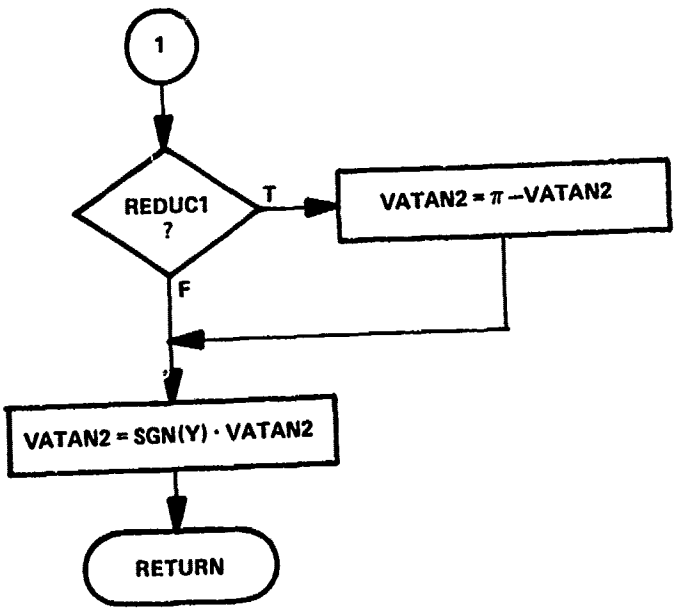
Path	Precision in Decimal Digits (relative)	Index in Reference
1	7.69	4940
2	11.39	4942
3	13.20	4943
4	15.01	4944

Inputs, Outputs & Internal Variables

XARG YARG	Two input arguments
ATNPTH	Selects "algorithm"
VATAN2	Returns value


VATAN2 (Y,X)
REDUC1 = .F.
REDUC2 = .F.
REDUC3 = .F.
NEGATIVE X TO POSITIVE
|Y| > |X|
ARG > TAN (15°)
X ?
X < 0
X = 0
X > 0
Y = 0
Y
N
REDUC1 = T.
|Y| > |X|
REDUC2 = T.
ARG = ABS(X/Y)
ARG = ABS (Y/X)
SIGNAL ERROR BY CALLING ATAN2 (0.,0.) RETURN
VA2.1


1
ARG ≤ TANI5
Y
ARG = [√3-1] - ARG -1 +ARG / ARG +√3
REDUC3 = .T.
N
COMPUTE VATAN2 USING POLYNOMINAL EXPANSION
REDUC3
T
VATAN2 = VATAN2 + π/6
F
REDUC2
T
VATAN2 = π/2 - VATAN2
F
VA3.1


1
REDUC1
?
T
VATAN2 = π –VATAN2
F
VATAN2 = SGN(Y) · VATAN2
RETURN

VSQRT

2.3.46 Function & Equations

The purpose of VSQRT is to compute the square root of a single argument. VSQRT implements the square root routine that would be implemented in an airborne computer. The algorithm to be used is selected by specifying SQRPTH to be 1, 2, 3 or 4: these algorithms are designed to produce, respectively, at least 24, 32, 40 or 48 bits of accuracy in the result. The only error condition is if the argument is less than zero; in such a case the standard Fortran square root error routine is invoked. An assembly language subroutine, VSQSUP is used to perform range reduction, as explained hereinafter.

I) Algorithms are generally of the form
   1) Make an initial approximation
   2) Apply some number of iterations of Newtons method

Steps 1) and 2) are related in that the better the initial approximation, the fewer the number of iterations are needed.

In the most general sense, the selection of a "optimal" routine is a function of:

a) space vs. speed considerations
b) relative speeds of add, multiply, divide
c) required precision

II) Initial approximations

The first step in making an initial approximation is to reduce the range of the argument (eg. from 10** -65 --->1.) The stated reduction seems the most likely to be implemented (on an avionics system) since alternatives rely on additional program instructions and constants. This range reduction (on a binary floating point computer) could be implemented by simply examining the exponent of the original argument. Stated mathematically:

REDUCT = GII (EXPO/2), where EXPO is exponent of NEWARG and GII is "greatest integer in" eg. GII (3/2) = 1, GII (-3/2) = -2

```
NEWARG  =  OLDARG/2**(REDUCT*2)
FACTOR  =  2**REDUCT
SQRT (OLDARG)  = SQRT (NEWARG) * FACTOR
```

Eight initial approximations (over the .25 ---> 1 range are given in the reference ("Computer Approximations", John F. Hart et al QA297, C738).

An iteration of Newtons method consists of:

```
NEWEST  = .5 * (OLDEST + NEWARG/OLDEST)
```

where OLDEST and NEWEST are respectively) the old and new estimates of the square root of NEWARG. Assuming that the multiplicatoion by .5 is done with a multiply, execution time is therefore assumed to be 14 (see below). From theory, and from results tabulated in the reference, one iteration tends to double the precision of the estimate.

Accordingly certain approximations can be eliminated from consideration

Eliminate (Hart's Index)	Since	Accuracy	Speed	Storage
292	293	Worse	Same	Same
290	0010 + 1 iteration	Worse	Same	Worse
291	0011 + 1 iteration	Worse	Same	Worse

Selection of an initial approximation (from the remaining five) is based on the required precision of the routine taken as a whole. Examine the following "Chart of Accuracy and Timings".

Assuming the following relative execution times for add, multiply and divide instructions.

CHART OF ACCURACY AND TIMINGS

INDEX	NUMBER OF ITERATIONS								
	0	1	2	3	4	5	6	7	8
0010	1.5 / 4	3.3 / 18	6.9 / 32	14.2 / 46					
0011	2.3 / 8	4.9 / 22	10.1 / 36	24.5 / 50					
0012	3.0 / 12	6.3 / 26	12.9 / 40	26.1 / 54					
0013	3.6 / 16	7.5 / 30	15.3 / 44	30.9 / 58					
0293	4.7 / 24	9.9 / 38	20.1 / 52						

X / Y

X DIGIT ACCURACY
Y EXECUTION TIME

Figure 6. Square Root Chart of Accuracy and Timings

ADD	1
MUL	3
DIV	10

the approximations take the following times and yield the following accuracies.

Approximation (Harts Index)	Execution Time	Precision in Decimal Digits (relative error)
0070	4	1.53
0071	8	2.30
0072	12	2.98
0073	16	3.60
0290	18	2.60
0291	22	3.66
0292	24	4.55
0293	24	4.73

We are targetting 48, 40, 32 and 24 bits of precision which correspond, respectively, to 14.4, 12.0, 9.6 and 7.2 decimal digit

We will therefore define the following paths

Path	Index	Number of Iterations	Approximate Precision (Decimal/Binary)	
1	0073	1	7.5	24.7
2	0071	2	10.1	33.3
3	0072	2	12.9	42.6
4	0073	2	15.3	50.5

Now consider the effect of order of computation on results of Newton's iteration in square root module.

Newton's iteration can be performed in more than one manner. Consider

A) $Y_{n+1} = 1/2 \ (Y_n + \frac{X}{Y_n})$

B) $Y_{n+1} = 1/2 \ (Y_n - \frac{X}{Y_n}) + \frac{X}{Y_n}$

C) $Y_{n+1} = Y_n + 1/2 \ (\frac{X}{Y_n} - Y_n)$

Examine the following two cases: (note that B' xxxx xxxx' means the binary fraction .xxxx xxxx).

Case I

$X = B'\ 1000\ 0000\ ' * 2^1$ (that is '1')

$Y_n = B'\ 1000\ 0001\ ' * 2^1$ (that is >'1')

$X/Y_n = B'\ 1000\ 0000\ ' * 2^1$ (with rounding)

A) = $B'\ 1000\ 0000\ ' * 2^1$

B) = $B'\ 1000\ 0000\ ' * 2^1$

C) = $B'\ 1000\ 0001\ ' * 2^1$

Case II

$X = B'\ 1000\ 0000\ ' * 2^1$ (that is '1')

$Y_n = B'\ 1111\ 1111\ ' * 2^0$ (that is <'1')

$X/Y_n = '1000\ 0001\ ' * 2^1$

A) = B' 1000 0001'

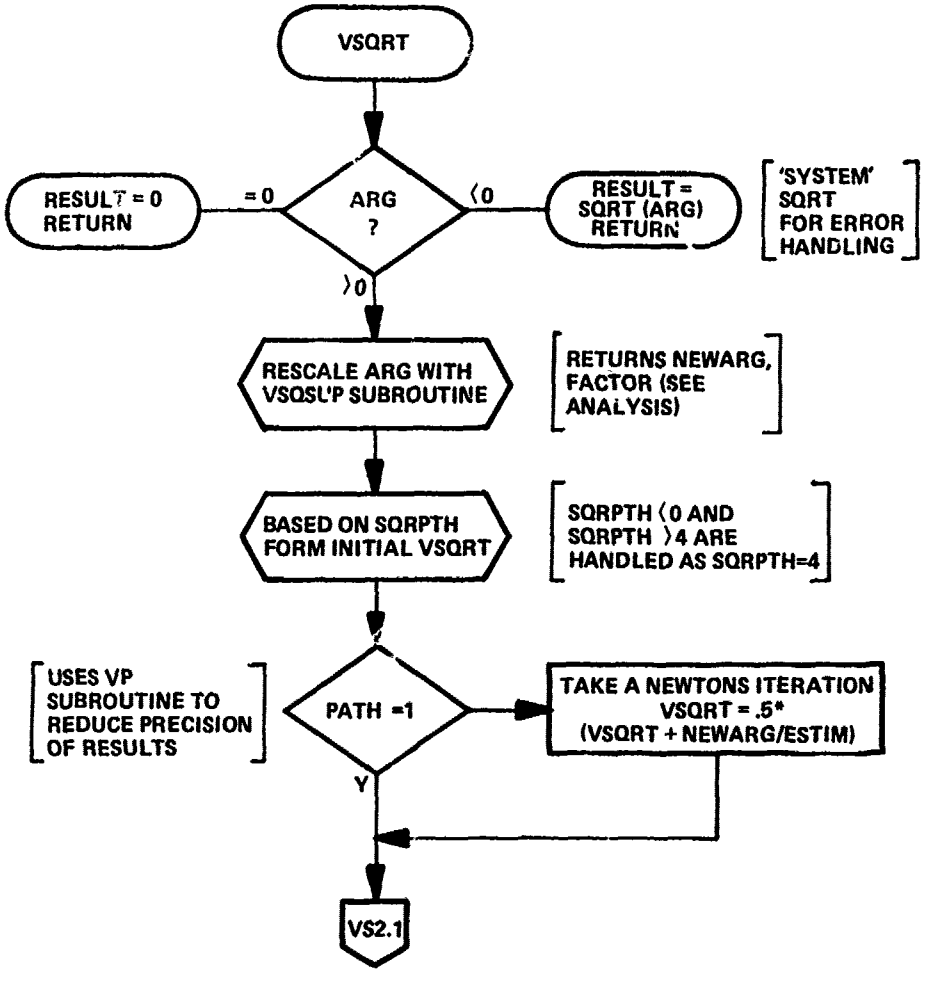
B) = B' 1000 0000'

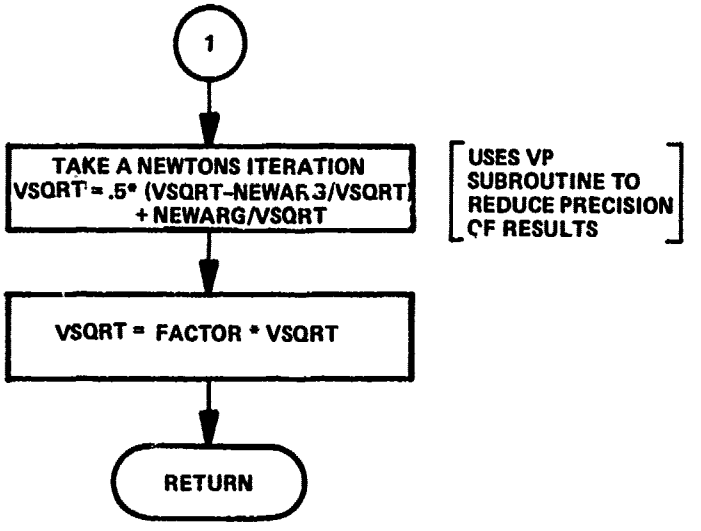
C) = B' 1000 0000'

Therefore, the last iteration will be done using "order" B.

Inputs, Outputs & Internal Variables

ARG - input argument
VSQRT - result value
SQRPTH- selects "algorithm" to be used


VSQRT
ARG ?
=0
RESULT = 0 RETURN
<0
RESULT = SQRT (ARG) RETURN
'SYSTEM' SQRT FOR ERROR HANDLING
>0
RESCALE ARG WITH VSQSU'P SUBROUTINE
RETURNS NEWARG, FACTOR (SEE ANALYSIS)
BASED ON SQRPTH FORM INITIAL VSQRT
SQRPTH <0 AND SQRPTH >4 ARE HANDLED AS SQRPTH=4
PATH =1
USES VP SUBROUTINE TO REDUCE PRECISION OF RESULTS
TAKE A NEWTONS ITERATION VSQRT = .5* (VSQRT + NEWARG/ESTIM)
Y
VS2.1


1
TAKE A NEWTONS ITERATION
VSQRT = .5* (VSQRT-NEWARG/VSQRT)
+ NEWARG/VSQRT
USES VP
SUBROUTINE TO
REDUCE PRECISION
OF RESULTS
VSQRT = FACTOR * VSQRT
RETURN

VSQSUP

2.3.47 Function & Equations

The purpose of this routine is to perform the 'range reduction' required by the square root routine. This range reduction consists of expressing an input argument, X, as

$X = Y * 2^{2n}$ where Y is between .25 and 1 and
n is a positive or negative integer or zero.

For a more complete description, refer to both the VSQRT documentation and the comments in the listing of the routine.

Inputs, Outputs & Internal Variables

First argument - 'X' above (input)
Second argument - 'Y' above (output)
Third argument - 2^n above (output)

VRTZYX

2.3.48 Function & Equations

This routine implements the ROTZYX routire that would be implemented in an airborne computer. It differs from the ROTZYX routine only in that it contains calls to the VP subroutine.

2.3.49 Function & Equations

This routine implements the ROTXYZ routine that would be implemented in an airborne computer. It differs from the ROTXYZ routine only in that it contains calls to the VP subroutine.

VMTM

2.3.50 Function & Equations

This routine implements the "matrix transpose by matrix" multiply routine that would be implemented in an airborne computer. It differs from the MTM routine only in that it contains calls to the VP subroutine.

VMATVC

2.3.51 Function & Equations

This routine implements the MATVEC routine that would be implemented in an airborne computer. It differs from the MATVEC routine only in that it contains calls to the VP subroutine.

2.3.52 Function & Equations

This routine implements the matrix multiply routine that would be implemented in an airborne computer. It differs from the MM routine only in that it contains calls to the VP sub-routine.

3.0 PROGRAM OPERATION

3.1 Introduction

The simulator is operated by providing it with PROFGEN's output tape, and by specifying certain inputs. The files required are described in an appendix titled "Local Files", the inputs are described in the next section, and sample SCOPE control cards are given in Appendix G.

3.2 Operator Inputs

a) Inputs

The operator input required consists of

i) First a title card, then

ii) One or more cards comprising a FORTRAN NAMELIST input for SIMPAR. This controls all the parameters of the simulation. The SIMPAR data input is described in the following section, and further explained in Appendix C.

The operator input might therefore look something like the following sample:

```
ITRNAV = 4 $
$SIMPAR    PRNT = .5, IPC = 27 * 0, 3 * 1,
THIS IS A TITLE
```

b) NAMELIST items & Defaults

The following variables are on the SIMPAR Namelist and can be input, at initialization, into the system.

Variable	Units	Default	Description
AQUANT	radians	0.0	Quantization units for attitude
TQUANT	radians	0.0	Quantization units for torquing angle for a nav cycle
VQUANT	ft/sec	0.0	Quantization units for delta velocity
START	sec	-1.E6	Time (on input tape) at which to start simulation
STOP	sec	1.E6	Time (on input tape) at which to stop simulation
PLOTIM	sec	1.E6	Simulated time between binary outputs
PRNT	sec	1.E6	Simulated time between printouts
RSTRT	sec	1.E6	See "Restart Hook" section
TOLJRK	ft/sec^3	1.E6	CMINTG's tolerance for 'jerk' see CMINTG documentation
HSOTIM	sec	1.E6	Time (on input tape) at which one should start (ortho) normalization of body-to-computational transformation.
HSOINC	sec	1.E6	Simulated time between (ortho) normalizations described under HSOTIM
SSX0	radians	0.0	Euler rotation about X of transformation from space stable to initial earth fixed
SSY0	radians	0.0	Euler rotation about Y of transformation from space stable to initial earth fixed
SSZ0	radians	0.0	Euler rotation about Z of transformation from space stable to initial earth fixed

Variables	Units	Default	Description
ITRNAV	CPS	0	Rate associated with navigation cycle; default is ITRATT
ITRATT	CPS	0	Rate associated with attitude cycle; default is 1./DELT
INCOR	none	100	Number of navigation cycles between orthonormalization of transformation from computational to earth fixed
IMUTYP	none	1	IMU type 1 is local level, 2 is space stable, 3 is strapdown
IERLIM	none	10	Error limit
IPC	none	0	See IPC list, variables description, for explanation
IOPNLP	none	0	Used by Local Level only. 0 implies closed loop, that is, IMU data is affected by misalignments caused by torquing errors. 1 implies IMU data remains unaffected.
CVD1	sec^{-1}	6.E-2	Vertical damping constant, see INRNAV description
CVD2	sec^{-2}	1.62E-3	Vertical damping constant, see INRNAV description
CVD3	sec^{-3}	1.62E-5	Vertical damping constant, see INRNAV description
ODX	radians	0.0	See description of ODX
ODY	radians	0.0	See description of ODY
ODZ	radians	0.0	See description of ODZ
GAINS	none	see listing	Gains for attitude filter; default based on values for 32 Hz rate

Variable	Units	Default	Description
RSMAX	radians	see listing	Residuals tolerance - see ATTFIL documentation
DRSMAX	radians	see listing	Delta-Residuals tolerance - see ATTFIL documentation
ITRFIL	CPS	32	Rate associated with attitude filter cycle
SQRPTH	none	4	Square root algorithm "path"
SINPTH	none	4	Sine algorithm "path"
COSPTH	none	4	Cosine algorithm "path"
ATNPTH	none	4	Arc tangent algorithm "path"
ROUND	none	1	1 Implies round, 0 implies truncate only
FRACTL	bits	48	Length of simulated fraction
EXPOL	bits	8	Length of simulated exponent

c) Using PROFGEN in Conjunction With Simulator

The reader is assumed to be familiar with PROFGEN and its usage. PROFGEN produces a tape which basically has the specific forces and attitude angles that would be sensed by an IMU. Several requirements should be kept in mind when running PROFGEN for use in conjunction with the simulator.

First, the output rate (DTO of the PROFGEN input) must be the same for all trajectory segments. This rate must be an integral multiple of the attitude rate one wishes to run in the simulator. It is suggested that these rates be limited to binary multiples of 1 CPS (cycle per second), that is 2, 4, 8, 16, 32, 64, 128 etc.

Second, local level wander azimuth should be specified for the PROFGEN run (LLMECH = 1).

Third, PROFGEN's RESTART should not be used; that is, RESTART should equal zero for all trajectory segments.

Fourth, PROFGEN's ERROR parameter must be small enough to keep PROFGEN's 'integration error' below the level one is trying to measure in the simulator. A value of 1.E-9 has proved satisfactory during testing.

3.3 Simulator Outputs

a) Binary Output File

The 'binary' output of the simulator (suitable for post processing) is written into "local file" TAPE30 (Fortran logical unit 30). The format of this output is described in an appendix. In general, it contains the differences between

i) the outputs of the simulator

ii) the same physical quantities as i) except that ii) was computed by PROFGEN.

b) Printed Output

The printed output of the simulator, shown in Figure 7, consists of three parts:

i) the parameters describing the trajectory segments by the simulator

ii) The parameters controlling the simulation

iii) Periodic printouts.

The next three exhibits are sample outputs, as follows:

i) a complete 8 sec local level run, with attitude filter, with printouts every half second. The units of the periodic printouts are as follows:

Degrees - Latitude, longitude, alpha, roll, pitch, yaw (yaw is actually heading), all nine filtered attitude parameters (yaw here includes alpha).

None - Gains pointers for each of filtered roll, filtered pitch and filtered yaw

Feet - Altitude

Feet/sec - VX (up), VY (east), VZ (north)

ii) the second page of a space stable run.

iii) the second page of a strapdown run.

Figure 7. NUMSIM Output Exhibits

```
PROFGEN PARAMETERS ARE:
IPROB:          33 NSEGT:
PPITCH: 0.0        ALFAO:   0.0          4 LLBEGH:                1 TSTART: 0.0          VTO:     950.00      PHEADO:  90.000
INITZ:          1 IPLOT:    1077952576 LATO:    45.000                LONO:   270.00      ALTO:    0.0         IPRNT:           1
                                       ROLRAT:  180.00

PARAMETERS FOR PROFILE NO.  1:
SEGLNT:  1.0000      RESTAR:
HEAD:    0.0         PITCH:   0.0          0 TURN:                  4 NPATH:             2 PACC:    1.5000      TACC:    0.0
HMIN:    0.10000D-07                       DTO:    0.78125D-02      MODE:                1 ERROR:   0.10000D-08 HMAX:    1000.0

PARAMETERS FOR PROFILE NO.  2:
SEGLNT:  2.0000      RESTAR:
HEAD:    0.0         PITCH:   1.5000       0 TURN:                  1 NPATH:             2 PACC:    0.0         TACC:    0.40000
HMIN:    0.10000D-07                       DTO:    0.78125D-02      MODE:                1 ERROR:   0.10000D-08 HMAX:    1000.0

PARAMETERS FOR PROFILE NO.  3:
SEGLNT:  2.0000      RESTAR:
HEAD:    0.0         PITCH:   -1.5000      0 TURN:                  1 NPATH:             2 PACC:    0.0         TACC:    0.40000
HMIN:    0.10000D-07                       DTO:    0.78125D-02      MODE:                1 ERROR:   0.10000D-08 HMAX:    1000.0

PARAMETERS FOR PROFILE NO.  4:
SEGLNT:  3.0000      RESTAR:
HEAD:    0.38000     PITCH:   0.0          0 TURN:                  2 NPATH:             1 PACC:    0.0         TACC:    0.50000
HMIN:    0.10000D-07                       DTO:    0.78125D-02      MODE:                1 ERROR:   0.10000D-08 HMAX:    1000.0
```

Figure 7. NUMSIM Output Exhibits (Continued)

```
IDEAL INS SIMULATOR          LOCAL LEVEL                                   00000070 PAGE   1

        I N I T I A L I Z A T I O N   P A R A M E T E R S

VERTICAL DAMPING:           CVD1: 0.60000D-01 SEC**-1,    CVD2: 0.16200D-02 SEC**-2,    CVD3:   0.16200D-04 SEC**-3
QUANTIZATION PARAMETERS:    AQUANT:   0.0          RAD    VQUANT:   0.0          FT/SEC TQUANT:   0.0          RAD
TIMES(SECS, EXCEPT INCOR):  START:   -.10000D 07          STOP:     0.10000D 07         PLOTIM:  0.10000D 07
                            PRNT.     0.50000          RSTRT:    0.10000D 07         INCOR:             100
TOLERANCES:                 TOLJRK:   0.10000D 07 FT/SEC IEBLIM:              5
IPC:                          0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                              0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,  1
LOCAL LEVEL PLATFORM MISALIGNMENTS, IOPMLP:          0
ATTITUDE FILTER TOLERANCES, RSMAX: 0.20000D-02   DRSMAX: 0.20000D-01 FIXED GAINS(COLUMNWISE):
 1.0000      0.0          0.0              1.0000      32.057      1.6575      0.99930      47.853      1019.2
0.95589      34.292       526.95          0.79591      16.511      137.16      0.67336      9.7338      57.194
0.54354      5.4080       21.418          0.46517      3.6090      10.938      0.37786      2.1314      4.5362
THIS VERSION DOES NOT IMPLEMENT VARIABLE PRECISION
 TIMING PARAMETERS
  TRAJECTORY SAMPLE RATE  128 CPS  TRAJECTORY PERIOD   0.00781250 SEC
  ATTITUDE CYCLE RATE     128 CPS  ATTITUDE PERIOD     0.00781250 SEC  TRAJ SAMP/ATT CYCLE   1
  NAV CYCLE RATE           32 CPS  NAV PERIOD          0.03125000 SEC  TRAJ SAMP/NAV CYCLE   4  ATT CYCLE/NAV CYCLE   4
  FILTER CYCLE RATE        32 CPS  FILTER PERIOD       0.03125000 SEC
 IMU STABILIZATION IS LOCAL LEVEL, WANDER AZIMUTH

 TIME        0.0
 LATTITUDE    45.0000000000        45.0000000000       0.355271367880D-14
 LONGITUDE   -90.0000000000       -90.0000000000       0.0
 ALPHA        0.0                  0.0                 0.0
 ALTITUDE     0.0                  0.0                 0.0
 VX (UP)      0.0                  0.0                 0.0
 VY (EAST)    950.000000000        950.000000000       0.0
 VZ (NORTH)  0.255137580629D-10    0.255137580629D-10  0.0
 ROLL         0.0                  0.0                 0.0
 PITCH        0.0                  0.0                 0.0
 YAW          90.0000000000        90.0000000000       0.248689957516D-13

 FLTRD ATTUD        ANGLE                 RATE                 ACCEL          PTR
 ROLL         0.0                  0.0                 0.0                   1
 PITCH        0.0                  0.0                 0.0                   1
 YAW          90.0000000000        0.0                 0.0                   1

 TIME        0.500000000000
 LATTITUDE    45.0000000000        45.0000000000       -.416378043155D-11
 LONGITUDE   -89.9981404418       -89.9981404418       -.142108547152D-13
 ALPHA       0.131490619409D-02   0.131490619412D-02   -.244974938772D-13
 ALTITUDE     0.0                 -.219712366132D-05    0.219712366132D-05
 VX (UP)      0.0                 -.464988161484D-05    0.464988161484D-05
 VY (EAST)    974.150000000        974.150000049       -.485537725581D-07
 VZ (NORTH)  0.261623613398D-10    0.243154726855D-05   -.243152110618D-05
 ROLL         0.0                 -.550750168990D-09    0.550750168990D-09
 PITCH        0.0                 -.322331689290D-06    0.322331689290D-06
 YAW          90.0000000000        90.0000000000       0.0

 FLTRD ATTUD        ANGLE                 RATE                 ACCEL          PTR
 ROLL        -.354014761482D-09   0.203815137129D-08   0.122701580001D-07    10
 PITCH       -.101904066161D-06   0.287999307780D-05   0.137751671161D-04    10
 YAW          90.0000000000       0.206528492092D-11   0.597015587409D-11    10

 TIME         1.00000000000
 LATTITUDE    45.0000000000        45.0000000000       -.826005930321D-11
 LONGITUDE   -89.9962342050       -89.9962342050       -.216715534407D-12
 ALPHA       0.266281915438D-02   0.266281915455D-02   -.168721211048D-12
```

Figure 7. NUMSIM Output Exhibits (Continued)

ALTITUDE	0.0	-.446438356762D-05	0.446438356762D-05
VX (UP)	0.0	-.483669523550D-05	0.483669523550D-05
VY (EAST)	998.300000000	998.300000172	-.171915417013D-06
VZ (NORTH)	0.268114020568D-10	0.243107539184D-05	-.243104858044D-05
ROLL	0.0	-.104995382607D-08	0.104995382607D-08
PITCH	0.0	-.580196943387D-06	0.580196943387D-06
YAW	90.0000000000	90.0000000000	-.206057393370D-12

FLTBD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.961250555317D-09	-.169982016402D-10	0.272663164331D-08	10
PITCH	-.480802452915D-06	0.590591887722D-06	0.308179926794D-05	10
YAW	90.0000000000	0.129061489030D-11	0.161342063415D-11	10

Figure 7. NUMSIM Output Exhibits (Continued)

IDEAL INS SIMULATOR LOCAL LEVEL 00000070 PAGE 2

	TRAJECTORY	SIMULATOR	DIFFERENCE
TIME	1.50000000000		
LATTITUDE	45.0000000000	45.0000000000	-.836308799990D-11
LONGITUDE	-89.9943046424	-89.9943046424	0.234230412843D-10
ALPHA	0.402722599949D-02	0.402722598294D-02	0.165482107184D-10
ALTITUDE	1.60999441667	1.60990207839	0.923382753935D-04
VX (UP)	6.43995533336	6.43995282507	0.250828566495D-05
VY (EAST)	998.279227959	998.279229276	-.131660266334D-05
VZ (NORTH)	0.268109701107D-10	-.360344558359D-08	0.363025655370D-08
ROLL	0.577087708476D-21	-.152061475751D-08	0.152061475751D-08
PITCH	0.369613162440	0.369612663141	0.499299585441D-06
YAW	90.0000000000	90.0000000000	0.262687649411D-10

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.150458176041D-08	-.870178279463D-09	0.198546153529D-09	10
PITCH	0.375335707889	0.906270890173	0.980851106245	10
YAW	90.0000000000	-.653547481195D-10	-.734820615933D-10	10

TIME	2.00000000000		
LATTITUDE	45.0000000000	45.0000000000	-.834887714518D-11
LONGITUDE	-89.9923751603	-89.9923751604	0.466258143206D-10
ALPHA	0.539157585546D-02	0.539157582251D-02	0.329502831270D-10
ALTITUDE	6.43991066703	6.43972613526	0.184531767226D-03
VX (UP)	12.8796426691	12.8796377139	0.495523004229D-05
VY (EAST)	998.216912702	998.216914214	-.151230170786D-05
VZ (NORTH)	0.268085692534D-10	-.611371350767D-08	0.614052207693D-08
ROLL	-.828026594582D-21	-.202462106900D-08	0.202462106900D-08
PITCH	0.739226324881	0.739225826345	0.498536183488D-06
YAW	90.0000000000	89.9999999999	0.588897819398D-10

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.202514881717D-08	-.102227518976D-08	-.611239714872D-10	10
PITCH	0.747873615046	0.842727057192	0.308945024537	10
YAW	89.9999999999	-.764439417166D-10	-.417543866398D-10	10

TIME	2.50000000000		
LATTITUDE	45.0000000000	45.0000000000	-.834177171782D-11
LONGITUDE	-89.9904458394	-89.9904458395	0.694377888522D-10
ALPHA	0.675581173557D-02	0.675581168649D-02	0.490775926842D-10
ALTITUDE	14.4895477550	14.4892725317	0.275223231531D-03
VX (UP)	19.3187940207	19.3187866811	0.733958611931D-05
VY (EAST)	998.113056821	998.113058529	-.170772557340D-05
VZ (NORTH)	0.268050720509D-10	-.844553507684D-08	0.847234014889D-08
ROLL	-.237660535349D-20	-.252877454014D-08	0.252877454014D-08
PITCH	1.10883948732	1.10883898999	0.497327348192D-06
YAW	90.0000000000	89.9999999999	0.977173897354D-10

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.252925980590D-08	-.101384822442D-08	-.147875619490D-10	10
PITCH	1.11048810234	0.754949551068	0.364294012196D-01	10
YAW	89.9999999999	-.840341271730D-10	-.267552447069D-10	10

Figure 7. NUMSIM Output Exhibits (Continued)

IDEAL INS SIMULATOR LOCAL LEVEL 00000070 PAGE 3

	TRAJECTORY	SIMULATOR	DIFFERENCE
TIME	3.00000000000		
LATTITUDE	45.0000000000	45.0000000000	-.832400814943D-11
LONGITUDE	-89.9885167603	-89.9885167604	0.918412013107D-10
ALPHA	0.811987665795D-02	0.811987659302D-02	0.6492998239770-10
ALTITUDE	25.7585706962	25.7582062736	0.3644225798390-03
VX (UP)	25.7571414241	25.7571317620	0.966209359277D-05
VY (EAST)	997.967664639	997.967666542	-.190269327049D-05
VZ (NORTH)	0.268003813586D-10	-.105988223892D-07	0.106256227705D-07
ROLL	-.444717694345D-20	-.303312780699D-08	0.303312780699D-08
PITCH	1.47845264976	1.47845215409	0.495673131207D-06
YAW	90.0000000000	89.9999999999	0.142769351896D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.303318645341D-08	-.100940007583D-08	-.168253177974D-11	10
PITCH	1.47855676966	0.739790637980	0.236795392522D-03	10
YAW	89.9999999999	-.259337517891D-10	-.251256261820D-10	10

TIME	3.50000000000		
LATTITUDE	45.0000000000	45.0000000000	-.834887714518D-11
LONGITUDE	-89.9865876823	-89.9865876824	0.117935883281D-09
ALPHA	0.948394081204D-02	0.948394072867D-02	0.833729552410D-10
ALTITUDE	37.0275936374	37.0273171277	0.276509696938D-03
VX (UP)	19.3187940207	19.3187867511	0.726968372433D-05
VY (EAST)	998.113056821	998.113056913	-.914214979275D-07
VZ (NORTH)	0.268035871276D-10	0.203240780039D-07	-.203972744168D-07
ROLL	-.319884388177D-20	-.350334701288D-08	0.350334701288D-08
PITCH	1.10883948732	1.10883898920	0.498123907899D-06
YAW	90.0000000000	89.9999999998	0.150567558421D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.350003295254D-08	-.980440275495D-09	-.298900484360D-10	10
PITCH	1.09738071663	-1.07348937020	-1.96222101894	10
YAW	89.9999999998	-.821859080756D-12	0.940360712586D-10	10

TIME	4.00000000000		
LATTITUDE	45.0000000000	45.0000000000	-.838085156829D-11
LONGITUDE	-89.9846583643	-89.9846583644	0.143977274547D-09
ALPHA	0.108481746646D-01	0.108481745646D-01	0.101791208275D-09
ALTITUDE	45.0772307254	45.0770523887	0.178336647931D-03
VX (UP)	12.8796426691	12.8796381095	0.455955322387D-05
VY (EAST)	998.216912702	998.216912877	-.175128263891D-06
VZ (NORTH)	0.268058353292D-10	0.178607127282D-07	-.178339068929D-07
ROLL	-.206096140085D-20	-.400685173750D-08	0.400685173749D-08
PITCH	0.739226324881	0.739225825745	0.499135318049D-06
YAW	90.0000000000	89.9999999998	0.152716950197D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.400770[illegible]18165D-08	-.102007102597D-08	-.480788652367D-10	10
PITCH	0.74[illegible]7247161	-.946256005256	-.617948811592	10
YAW	89.9999999998	0.108008055801D-10	0.502691737337D-10	10

Figure 7. NUMSIM Output Exhibits (Continued)

IDEAL INS SIMULATOR LOCAL LEVEL 00000070 PAGE 4

	TRAJECTORY	SIMULATOR	DIFFERENCE	
TIME	4.50000000000			
LATTITUDE	45.0000000000	45.0000000000	-.839861513668D-11	
LONGITUDE	-89.9827288862	-89.9827288864	0.169880109979D-09	
ALPHA	0.122125216581D-01	0.122125215380D-01	0.120102999594D-09	
ALTITUDE	49.9071469757	49.9070652778	0.816979325045D-04	
VX (UP)	6.43995533336	6.43995341916	0.191419535600D-05	
VY (EAST)	998.279227959	998.279228218	-.258780119111D-06	
VZ (NORTH)	0.268070426968D-10	0.154726134920D-07	-.154458114493D-07	
ROLL	-.188705128821D-20	-.451045868604D-08	0.451045868604D-08	
PITCH	0.3696131624 40	0.369612662736	0.499704718326D-06	
YAW	90.0000000000	89.9999999999	0.148212109252D-09	

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.451078284022D-08	-.101080548718D-08	-.956891046463D-11	10
PITCH	0.366314345262	-.770672763972	-.728557234686D-01	10
YAW	89.9999999999	0.173040295043D-10	0.291839054732D-10	10

TIME	5.00000000000			
LATTITUDE	45.0000000000	45.0000000000	-.842348413244D-11	
LONGITUDE	-89.9807993282	-89.9807993284	0.195615967868D-09	
ALPHA	0.135769252197D-01	0.135769250814D-01	0.138307565054D-09	
ALTITUDE	51.5171413924	51.5171548086	-.134162483079D-04	
VX (UP)	-.397903932026D-12	0.667131370247D-06	-.667131768151D-06	
VY (EAST)	998.300000000	998.300000342	-.342311011536D-06	
VZ (NORTH)	0.268072306985D-10	0.131599069975D-07	-.131330997605D-07	
ROLL	-.165004223358D-20	-.501422058224D-08	0.501422058224D-08	
PITCH	-.162745240308D-13	-.499832101747D-06	0.499832085473D-06	
YAW	90.0000000000	89.9999999999	0.137081457297D-09	

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	-.501425599876D-08	-.100798072984D-08	-.117694018165D-11	10
PITCH	-.209686845865D-03	-.740354398570	-.472052502541D-03	10
YAW	89.9999999999	0.292074117410D-10	0.263237586744D-10	10

TIME	5.50000000000			
LATTITUDE	44.9999960159	44.9999960137	0.212078177242D-08	
LONGITUDE	-89.9788697694	-89.9788697696	0.204561700912D-09	
ALPHA	0.149413292971D-01	0.149413291523D-01	0.144765192021D-09	
ALTITUDE	51.5171413924	51.5171605473	-.191549159005D-04	
VX (UP)	-.384635292042D-12	0.818412180028D-06	-.818412564663D-06	
VY (EAST)	998.279268080	998.279266199	0.188053741113D-05	
VZ (NORTH)	-6.43373310617	-6.43485593507	0.112282889715D-02	
ROLL	11.5830727801	11.5830728495	-.693255770567D-07	
PITCH	-.162754051762D-13	-.499999822436D-06	0.499999806161D-06	
YAW	90.3692560406	90.3692560404	0.144179779227D-09	

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	11.5830728495	0.0	0.0	1
PITCH	0.5746998105360-02	0.167393386378	0.981891502280	10
YAW	90.3809452169	1.01264219573	1.18583003127	10

Figure 7. NUMSIM Output Exhibits (Continued)

IDEAL INS SIMULATOR LOCAL LEVEL 00000070 PAGE 5

	TRAJECTORY	SIMULATOR	DIFFERENCE
TIME	6.00000000000		
LATTITUDE	44.9999868999	44.9999868973	0.267649724606D-08
LONGITUDE	-89.9769402411	-89.9769402413	0.200543581741D-09
ALPHA	0.163057116316D-01	0.163057114896D-01	0.141968481242D-09
ALTITUDE	51.5171413924	51.5171603803	-.189879485255D-04
VX (UP)	-.366710337323D-12	0.806420773499D-06	-.806421140209D-06
VY (EAST)	998.277728699	998.277723595	0.510444033353D-05
VZ (NORTH)	-6.66831187426	-6.66982928454	0.151741027648D-02
ROLL	0.222363324577D-11	0.594959509546D-07	-.594937273214D-07
PITCH	-.162764650310D-13	-.500002541657D-06	0.500002525380D-06
YAW	90.3827195914	90.3827195913	0.142716061191D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	0.595718353451D-07	-.334585747685D-08	0.636826258335D-08	10
PITCH	0.865330442475D-02	0.103557285815	0.309062810117	10
YAW	90.3877044924	-.481745739162D-01	-.704419499178	10

TIME	6.50000000000		
LATTITUDE	44.9999777392	44.9999777344	0.475730033145D-08
LONGITUDE	-89.9750107136	-89.9750107138	0.210501838183D-09
ALPHA	0.176700932338D-01	0.176700930847D-01	0.149104475294D-09
ALTITUDE	51.5171413924	51.5171602135	-.188211017900D-04
VX (UP)	-.366710337323D-12	0.796870516843D-06	-.796870883553D-06
VY (EAST)	998.277570161	998.277565082	0.507926199589D-05
VZ (NORTH)	-6.69200370850	-6.69352113550	0.151742700079D-02
ROLL	0.222363324577D-11	0.571531549270D-07	-.571509312937D-07
PITCH	-.162764650310D-13	-.499993994925D-06	0.499993978648D-06
YAW	90.3840793755	90.3840793754	0.151089807332D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	0.572034754091D-07	-.410170111677D-08	0.170479847744D-08	10
PITCH	0.164883835241D-02	0.157232187868D-01	0.364232416283D-01	10
YAW	90.3764294789	-.943662836169D-01	-.304706600096	10

TIME	7.00000000000		
LATTITUDE	44.9999685459	44.9999685391	0.683812118041D-08
LONGITUDE	-89.9730811866	-89.9730811869	0.220488516334D-09
ALPHA	0.190344741825D-01	0.190344740262D-01	0.156304955957D-09
ALTITUDE	51.5171413924	51.5171600469	-.186545027745D-04
VX (UP)	-.366710337323D-12	0.787386[illegible]78038D-06	-.787386344749D-06
VY (EAST)	[illegible]277411060	998.277406006	0.505408485196D-05
VZ (NORTH)	[illegible]9552759	-6.71721297066	0.151744307264D-02
ROLL	0.222363324577D-11	0.548103365711D-07	-.548081129378D-07
PITCH	-.162764650310D-13	-.499985371093D-06	0.499985354817D-06
YAW	90.3854391590	90.3854391589	0.159435131764D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	0.548190798815D-07	-.460381505256D-08	0.186541536472D-09	10
PITCH	0.104027163710D-03	0.563217831300D-03	0.233722563300D-03	10
YAW	90.3836865542	-.146075108528D-01	-.417046992289D-01	10

Figure 7. NUMSIM Output Exhibits (Continued)

IDEAL INS SIMULATOR LOCAL LEVEL 00000070 PAGE 6

	TRAJECTORY	SIMULATOR	DIFFERENCE
TIME	7.50000000000		
LATTITUDE	44.9999593201	44.9999593112	0.891896689836D-08
LONGITUDE	-89.9711516603	-89.9711516605	0.230500063481D-09
ALPHA	0.2039885447560-01	0.203988543120D-01	0.163569814811D-09
ALTITUDE	51.5171413924	51.5171598806	-.184881766465D-04
VX (UP)	-.366710337323D-12	0.777967621592D-06	-.777967988303D-06
VY (EAST)	998.277251398	998.277246369	0.502890742382D-05
VZ (NORTH)	-6.73938733149	-6.74090478998	0.151745849202D-02
ROLL	0.222363324577D-11	0.524674967854D-07	-.524652731521D-07
PITCH	-.162764650310D-13	-.499976670169D-06	0.499976653893D-06
YAW	90.3867989419	90.3867989417	0.167805325191D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	0.524679933567D-07	-.458333396641D-08	-.302420341307D-12	10
PITCH	-.127516744191D-04	-.174671148270D-03	-.520851801369D-03	10
YAW	90.3866605089	0.181513994693D-02	-.103287173369D-02	10

TIME	8.00000000000		
LATTITUDE	44.9999500619	44.9999500509	0.109998374853D-07
LONGITUDE	-89.9692221346	-89.9692221348	0.240540032337D-09
ALPHA	0.217632341106D-01	0.217632339397D-01	0.170898973792D-09
ALTITUDE	51.5171413924	51.5171597145	-.183221480405D-04
VX (UP)	-.366710337323D-12	0.768615891605D-06	-.768616258315D-06
VY (EAST)	998.277091173	998.277086169	0.500372453871D-05
VZ (NORTH)	-6.76307912014	-6.76459659340	0.151747325892D-02
ROLL	0.222363324577D-11	0.501246364683D-07	-.501224128351D-07
PITCH	-.162764650310D-13	-.499967892153D-06	0.499967875876D-06
YAW	90.3881587240	90.3881587238	0.176136438768D-09

FLTRD ATTUD	ANGLE	RATE	ACCEL	PTR
ROLL	0.501245663750D-07	-.468069868341D-08	-.285789062979D-11	10
PITCH	-.351032574763D-05	-.282962983600D-04	-.583223190313D-04	10
YAW	90.3881684260	0.287923298835D-02	0.512932504488D-03	10

***** END OF INPUT TAPE: ERROR LEVEL 10

***** SIMULATION TERMINATED: ERROR SEVERITY OF 10: ERROR LIMIT IS 5

Figure 7. NUMSIM Output Exhibits (Continued)

```
VERTICAL DAMPING              CVD1  0.60000D-01 SEC**-1,    CVD2  0.16200D-02 SEC**-2,    CVD3    0.16200D-04 SEC**-3
QUANTIZATION PARAMETERS       AQUANT   0.0          RAD     VQUANT    0.0          FT/SEC  TQUANT   0.0          RAD
TIMES(SECS, EXCEPT INCOR)     START    -.10000D 07          STOP      0.10000D 07          PLOTIM   0.10000D 07
                              PRNT     0.50000              RSTRT     0.10000D 07          INCOR              100
TOLERANCES                    TOL RP   0.10000D 07 FT/SEC   TEMLIM              5
IPC                             0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                                0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,  1
SPACE STABLE PLATFORM INITIAL ORIENTATION AS EULER ANGLES(RADIANS)  SSX0      0.0                  SSY0      0.0
                                                                   SSZ0      0.0

THIS VERSION DOES NOT IMPLEMENT VARIABLE PRECISION
 TIMING PARAMETERS
  TRAJECTORY SAMPLE RATE   128 CPS  TRAJECTORY PERIOD    0.00781250 SEC
  ATTITUDE CYCLE RATE      128 CPS  ATTITUDE PERIOD      0.00781250 SEC  TRAJ SAMP/ATT CYCLE   1
  NAV CYCLE RATE            16 CPS  NAV PERIOD           0.06250000 SEC  TRAJ SAMP/NAV CYCLE   8  ATT CYCLE/NAV CYCLE   8
 IMU STABILIZATION IS SPACE STABLE

 TIME          0.0
 LATTITUDE      45.0000000000          45.0000000000        0.355271367880D-14
 LONGITUDE     -90.0000000000         -90.0000000000        0.0
 ALPHA          0.0                    0.0                  0.0
 ALTITUDE       0.0                    0.0                  0.0
 VX (UP)        0.0                    0.0                  0.0
 VY (EAST)       950.000000000          950.000000000       0.0
 VZ (NORTH)  0.255137580629D-10     0.255137580629D-10       0.0
 ROLL           0.0                    0.238541601110D-14   -.238541601110D-14
 PITCH          0.0                    0.298034568272D-14   -.298034568272D-14
 YAW             90.0000000000          90.0000000000        0.142108547152D-13

 TIME          0.500000000000
 LATTITUDE      45.0000000000          45.0000000000        -.162607705079D-10
 LONGITUDE     -89.9981404418         -89.9981404418        -.284217094304D-13
 ALPHA          0.131490619409D-02     0.131490619412D-02   -.292885290673D-13
 ALTITUDE       0.0                   -.847515227725D-05    0.847515227725D-05
 VX (UP)        0.0                   -.183819937419D-04    0.183819937419D-04
 VY (EAST)       974.150000000          974.150000047       -.474794887050D-07
 VZ (NORTH)  0.261623613398D-10     0.972549679732D-05      -.972547063496D-05
 ROLL           0.0                    0.906537598085D-10   -.906537598085D-10
 PITCH          0.0                   -.923232690699D-04    0.923232690699D-04
 YAW             90.0000000000          89.9999076767        0.923232690937D-04

 TIME           1.00000000000
 LATTITUDE      45.0000000000          45.0000000000        -.326600968492D-10
 LONGITUDE     -89.9962342050         -89.9962342050        -.177635683940D-12
 ALPHA          0.266281915438D-02     0.266281915452D-02   -.142532307571D-12
 ALTITUDE       0.0                   -.173022925820D-04    0.173022925820D-04
 VX (UP)        0.0                   -.184770617289D-04    0.184770617289D-04
 VY (EAST)       998.300000000          998.300000108       -.108222536710D-06
 VZ (NORTH)  0.268114020568D-10     0.972246886403D-05      -.972244205263D-05
 ROLL           0.0                    0.107051904608D-09   -.107051904608D-09
 PITCH          0.0                   -.923232689599D-04    0.923232689599D-04
 YAW             90.0000000000          89.9999076767        0.923232689765D-04

 TIME           1.50000000000
 LATTITUDE      45.0000000000          45.0000000000        -.334736682817D-10
 LONGITUDE     -89.9943046424         -89.9943046425        0.964845980889D-10
 ALPHA          0.402722599949D-02     0.402722593128D-02   0.682123845255D-10
 ALTITUDE       1.60999441667          1.60964869837        0.345718296245D-03
 VX (UP)        6.43995533336          6.43994678947        0.854388323979D-05
 VY (EAST)      998.279227959          998.279231866        -.390707015185D-05
 VZ (NORTH)  0.268109701107D-10     -.186426510757D-07       0.186694620458D-07
 ROLL           0.577087708476D-21     0.107872345722D-09   -.107872345722D-09
 PITCH          0.369613162440         0.369520839103       0.923233373186D-04
 YAW             90.0000000000          89.9999076767
```

Figure 7. NUMSIM Output Exhibits (Continued)

```
VERTICAL DAMPING            CVD1  0.16000D-01 SEC**-1,    CVD2  0.16200D-02 SEC**-2,    CVD3   0.16200D-04 SEC**-3
QUANTIZATION PARAMETERS     AQUANT    0.0          RAD     VQUANT    0.0          FT/SEC  TQUANT   0.0          RAD
TIMES(SECS, EXCEPT INCOR)   START     -.10000D 07          STOP      0.10000D 07          PLOTIM   0.10000D 07
                            PRNT      0.50000              RSTRT     0.10000D 07          INCOR              100
TOLERANCES                  TOLJRK    0.10000D 07 FT/SEC   IERLIM              5
IPC                           0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                              0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1
HIGH SPEED STRAPDOWN NORMALIZATIONS WILL START AT TIME 0.10000D 07 AND HAPPEN EVERY 0.10000D 07 SECONDS
THIS VERSION DOES NOT IMPLEMENT VARIABLE PRECISION
 TIMING PARAMETERS
  TRAJECTORY SAMPLE RATE   128 CPS  TRAJECTORY PERIOD    0.00781250 SEC
  ATTITUDE CYCLE RATE      128 CPS  ATTITUDE PERIOD      0.00781250 SEC  TRAJ SAMP/ATT CYCLE   1
  NAV CYCLE RATE             4 CPS  NAV PERIOD           0.25000000 SEC  TRAJ SAMP/NAV CYCLE  32  ATT CYCLE/NAV CYCLE  32
 IMU STABILIZATION IS STRAPDOWN

  IDEAL INS SIMULATOR          STRAPDOWN EXAMPLE                                                          PAGE   2

              TRAJECTORY              SIMULATOR              DIFFERENCE

 TIME         0.0
 LATTITUDE     45.0000000000          45.0000000000        0.355271367880D-14
 LONGITUDE    -90.0000000000         -90.0000000000        0.0
 ALPHA         0.0                    0.0                  0.0
 ALTITUDE      0.0                    0.0                  0.0
 VX (UP)       0.0                    0.0                  0.0
 VY (EAST)     950.000000000          950.000000000        0.0
 VZ (NORTH)   0.255137580629D-10     0.255137580629D-10    0.0
 ROLL          0.0                    0.0                  0.0
 PITCH         0.0                    0.0                  0.0
 YAW           90.0000000000          90.0000000000        0.142108547152D-13

 TIME         0.500000000000
 LATTITUDE     45.0000000000          45.0000000002         .220300222509D-09
 LONGITUDE    -89.9981404418         -89.9981404418        -.383693077310D-12
 ALPHA         0.1314906194O9D-02     0.131490619437D-02   -.289790912327D-12
 ALTITUDE      0.0                   -.109009022338D-03     0.109009022338D-03
 VX (UP)       0.0                   -.294184310624D-03     0 294184310624D-03
 VY (EAST)     974.150000000          974.150000554        -.553912002488D-06
 VZ (NORTH)   0.261623700135D-10     0.155657118470D-03    -.155657092308D-03
 ROLL          0.0                   -.308947549983D-08     0.308947549983D-08
 PITCH         0.0                   -.412584569672D-05     0.412584569672D-05
 YAW           90.0000000000          90.0000000000        -.163424829225D-12

 TIME         1.00000000000
 LATTITUDE     45.0000000000          45.0000000005        -.482785367240D-09
 LONGITUDE    -89.9962342050         -89.9962342050        -.251887399827D-11
 ALPHA         0.266281915438D-02     0.266281915620D-02   -.181851473983D-11
 ALTITUDE      0.0                   -.251056012248D-03     0.251056012248D-03
 VX (UP)       0.0                   -.295776632208D-03     0.295776632208D-03
 VY (EAST)     998.300000000          998.300001540        -.153971882355D-05
 VZ (NORTH)   0.260114046589D-10     0.155658255229D-03    -.155658228417D-03
 ROLL          0.0                   -.250107015504D-08     0.250107015504D-08
 PITCH         0.0                   -.412584416106D-05     0.412584416106D-05
 YAW           90.0000000000          90.0000000000        -.164135371961D-11

 TIME         1.50000000000
 LATTITUDE     45.0000000000          45.0000000005        -.529031041197D-09
 LONGITUDE    -89.9943046424         -89.9943043704        -.271946774433D-06
 ALPHA         0.402722599949D 02     0.402741829496D 02   -.192295470548D-06
 ALTITUDE      1.60999441667          1.58801359164         0.219808250260D-01
 VX (UP)       6.43995533336          6.38959412529         0.503612080675D-01
 VY (EAST)     998.279227959          998.467967800        -.188739840498
 VZ (NORTH)   0.268109701107D-10    -.152765546088D-04     0.152765814198D-04
 ROLL          0.574706654656D-21     0.724526912646D-07   -.724526912646D-07
 PITCH         0.369613162440         0.369613360216       -.197775903543D-06
 YAW           90.0000000000          90.0000001928        -.192757632078D-06

                                                                                                           PAGE   3
```

None of the printouts that can be generated by DOUT are in these samples. The units of the DOUT printout can be found in the variable listings.

3.4 Operational Aids

a) Cautions and Other Miscellaneous Advice

1) Read the section on using PROFGEN; the simulator assumes (and does not check) all of the mentioned conditions.

2) PROFGEN has shown a tendency to do strange things with the low order bits of the TIME on the first few seconds, specifically a DTO of .0078125 may come out .007812499999 or .00781250001. This condition can be circumvented by specifying a slightly smaller number (.0078124 in our example) for quantities such as PRNT, PLOTIM, HSOINC.

3) The GAINS in the Block Data subprogram are designed for 32 CPS, and will work miserably at any other rate.

4) One cannot get printout or binary output at a rate higher than that the navigation rate.

5) No "range checking" is performed on the SIMPAR inputs; that is, setting IMUTYP to 4 or IPC (26) to 10 might produce unpredictable results.

b) Error Messages

The error messages that can be produced by the simulator is as follows:

1) Timing error message - see attached;generates error level 5

2) Tape read error message - (not illustrated) generates error level 5

3) End of input file - see attached generates error level 10

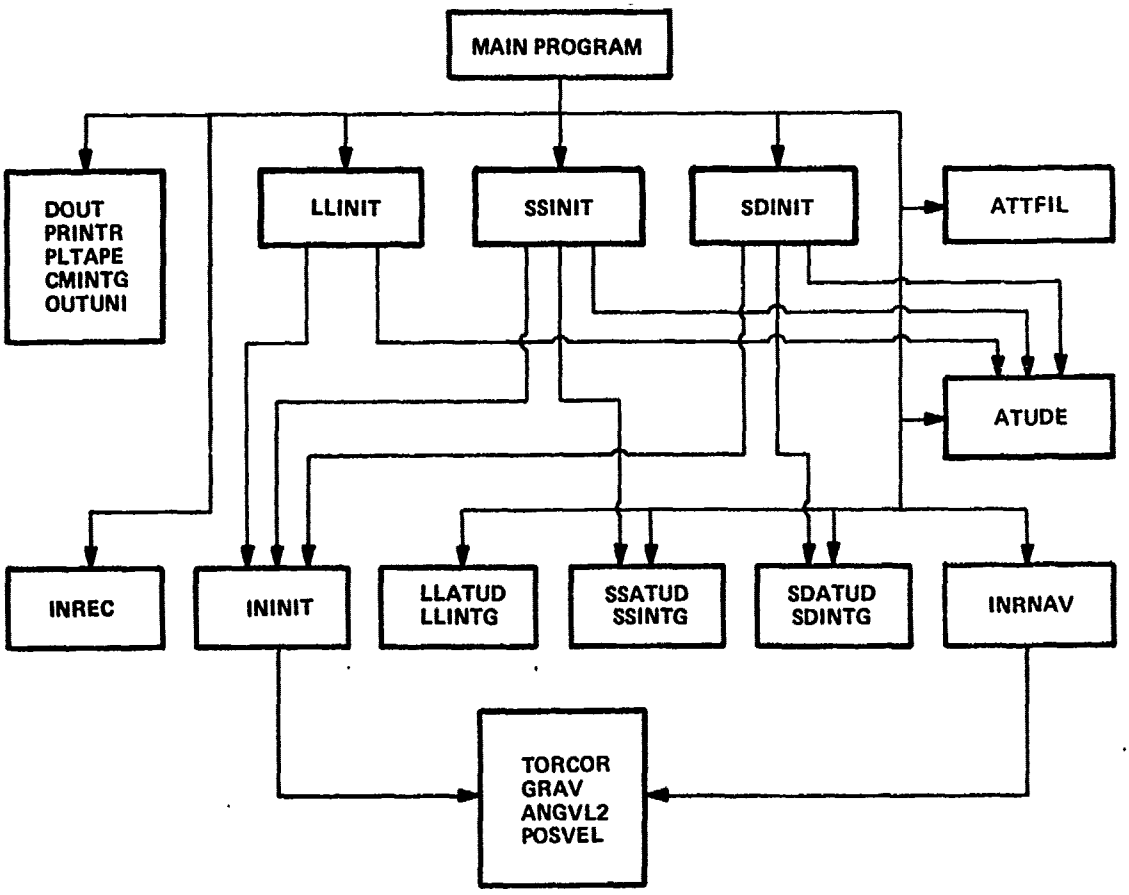
If the error level generated is greater than or equal to the program variable IERLIM (which can be modified by the SIMPAR input), program execution ceases after the end of the current navigation cycle as shown in Figure 8.

Figure 8. NUMSIM Error Message Sample

```
VERTICAL DAMPING:            CVD1: 0.60000D-01 SEC**-1,    CVD2. 0.16200D-02 SEC**-2,    CVD3    0.16200D-04 SEC**-3
QUANTIZATION PARAMETERS·     AQUANT:    0.0          RAD    VQUANT·    0.0          FT/SEC  TQUANT·   0.0          RAD
TIMES(SECS, EXCEPT INCOR).   START:     -.10000D 07         STOP:      0.10000D 07         PLOTIM·   0.10000D 07
                             PRNT,      0.10000D 07         RSTRT:     0.10000D 07         INCOR:              100
TOLERANCES:                  TOLJRK·    0.10000D 07 FT/SEC  IERLIM.              5
IPC·                           0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                               0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,  1
LOCAL LEVEL PLATFORM MISALIGNMENTS, IOPNLP.          0
ATTITUDE FILTER TOLERANCES, RSMAX· 0.20000D-02   DRSMAX  0.20000D-01 FIXED GAINS(COLUMNWISE)·
 1.0000      0.0          0.0              1.0000        32.057        1.6575          0.99930       47.853        1019.2
0.95689      34.292       526.95           0.79591       16.511        137.16          0.67336       9.7338        57.194
0.54354      5.4080       21.418           0.46517       3.6090        10.938          0.37786       2.1314        4.5362
0.33207      1.5407       2.6141
THIS VERSION DOES NOT IMPLEMENT VARIABLE PRECISION

 ***** TIMING ERROR:  DESIRED ATUD PERIOD IS 0.7874015719D-02 SEC;  TRAJECTORY SAMPLE PERIOD IS 0.7812500000D-02
       MUST BE EVEN MULTIPLES          ERROR LEVEL 5


 ***** TIMING ERROR:  DESIRED NAV PERIOD IS 0.7999997586D-02 SEC;  DESIRED ATTITUDE PERIOD IS 0.7812500000D-02
       MUST BE EVEN MULTIPLES          ERROR LEVEL 5


 ***** TIMING ERROR: EITHER ITRFIL IS NOT AN INTEGRAL NUMBER OF ATTITUDE CYCLES, OR
                     ITRNAV IS NOT AN INTEGRAL MULTIPLE OF ITRATT    ERROR LEVEL 5


 TIMING PARAMETERS
  TRAJECTORY SAMPLE RATE  128 CPS  TRAJECTORY PERIOD   0.00781250 SEC
  ATTITUDE CYCLE RATE     128 CPS  ATTITUDE PERIOD     0.00781250 SEC  TRAJ SAMP/ATT CYCLE   1
  NAV CYCLE RATE          128 CPS  NAV PERIOD          0.00781250 SEC  TRAJ SAMP/NAV CYCLE   1  ATT CYCLE/NAV CYCLE   1
  FILTER CYCLE RATE       128 CPS  FILTER PERIOD       0.00781250 SEC
 IMU STABILIZATION IS LOCAL LEVEL, WANDER AZIMUTH

 TIME         0.0
 LATTITUDE     45.0 00000000        45.0000000000      0.355271367880D-14
 LONGITUDE   -90.0.00000000       -90.0000000000      0.0
 ALPHA        0.0                  0.0                0.0
 ALTITUDE     0.0                  0.0                0.0
 VX (UP)      0.0                  0.0                0.0
 VY (EAST)     950.000000000        950.000000000     0.0
 VZ (NORTH) 0.255137580629D-10   0.255137580629D-10    0.0
 ROLL         0.0                  0.0                0.0
 PITCH        0.0                  0.0                0.0
 YAW           90.0000000000        90.0000000000     0.248689957516D-13

 FLTRD ATTUD        ANGLE                 RATE                  ACCEL        PTR
 ROLL         0.0                  0.0                0.0                   1
 PITCH        0.0                  0.0                0.0                   1
 YAW           90.0000000000       0.0                0.0                   1


 *****  SIMULATION TERMINATED:  ERROR SEVERITY OF  5. ERROR LIMIT IS  5
READY
```

APPENDICES

A. SUBROUTINE CALL STRUCTURE*



*BLOCK DATA SUBPROGRAM IS NOT EXPLICITLY CALLED, AND IS NOT SHOWN. LIBRARY SUBROUTINES (e.g., TRIGONOMETRIC FUNCTIONS, MATRIX MULTIPLY, ETC.) ARE, LIKEWISE, NOT SHOWN.

B. Delivery and Installation

Two versions of the simulator are being delivered. One has the "variable precision" feature implemented; the other does not. The former requires substantially (order of magnitude estimate: 10 times) more execution time than the latter.

Listing of both programs are presented in Volume IV.

The software is being delivered in sections as follows:

1)	NUMEXC	11)	SDFLT
2)	NUMIMU	12)	VUMEXC
3)	NUMFLT	13)	VUMFLT
4)	SUBS1	14)	VUBS1
5)	SUBS2	15)	VUBS3
6)	SSIMU	16)	VPREC
7)	SSFLT	17)	VSFLT
8)	LLIMU	18)	VLFLT
9)	LLFLT	19)	VDFLT
10)	SDIMU		

The makeup of each version can be seen in the attached table.

Note that

i) "Identical" implies the code for some section of the two simulators is identical

ii) "Same Names" implies the code differs only by the inclusion of the precision varying subroutines

iii) All sections whose names and with 'FLT' are flight code, as are SUBS1, SUBS2, VUBS1 and VUBS3.

Without Precision			With Precision	
NUMEX	Main program		VUMEXC	Main program
NUMIMU	BLOCK DATA PLTAPE INREC DOUT OUTUNI CMINTG PRINTR		NUMIMU	Identical
NUMFLT	TORCOR INRNAV POSVEL GRAV ANGVL2 ATUDE	ININIT ATTFIL	VUMFLT	Same names
SUBS1	ORTHO DCMUPD		VUBS1	Same names
SUBS2	MATVEC MTM MM ROTXYZ MATRAN ROTZYX	QNTIZ	SUBS2	Identical
			VUBS3	VRTXYZ VRTZYX VMTM VMATVC VMM
			VPREC	VP VPINIT VSIN VCOS VSINCO VATAN2 VSQRT VSQSUP
SSIMU	SSATUD SSINTG		SSIMU	Identical
SSFLT	SSINIT SSTFRM SSATT		VSFLT	Same names

Without Precision		With Precision	
LLIMU	LLATUD LLINTG	LLIMU	Identical
LLFLT	LLINIT LLATT	VLFLT	Same names
SDIMU	SDATUD SDINTG	SDIMU	Identical
SDFLT	SDINIT SDATT	VDFLT	Same names
	HSINTG HSINT2		

C. Sample Set of Simulator Inputs

Desired: strapdown system
attitude at 128 CPS
navigation at 4 CPS
attitude filter at 32 CPS
3rd order quaternion update
default rates, tolerances
Navigation DCM orthonomalize every nav cycle
No quantization
printout every 5 simulated seconds
binary outputs every .5 seconds
quaternion normalize every attitude cycle
second order navigation DCM update
higher 'order' algorithms, exact angular velocity

Assumes PROFGEN starting time was 0.

Card column 2

```
$SIMPAR   IMUTYP=3,ITRATT=128,ITRNAV=4,
ITRFIL=32,IPC=24*0,1,2,1,1,1,1,INCOR=1,
PRNT=5.,PLTIME=0.,PLOTIM=.5,HSOTIM=0,
HSOINC=.007812$
```

D. Assumed Format of Binary Input

Record	Contents
1	Date and Time from PROFGEN
2	PROFGEN's $PRDATA
3	PROFGEN's $PASDATA
4 thru n	14 word records (see variable listing)

1) TIME
2) LAT$T
3) LONG$T
4) ALF$T
5) HB
6) ETA$T (1)
7) ETA$T (2)
8) ETA$T (3)
9) V$T (1)
10) V$T (2)
11) V$T (3)
12) SF$T (1)
13) SF$T (2)
14) SF$T (3)

E. "Local Files" Required By Simulator Under CDC INTERCOM/SCOPE

TAPE5 - operator input to program
TAPE6 - printed output
TAPE20 - binary input file (input from PROFGEN)
TAPE30 - binary output file from simulator

F. Format of Binary Output

Record	Contents
1	80character title
2	11 words as follows

1) TIME - time
2) DLAT - latitude difference
3) DLONG - longitude difference
4) DALF - alpha difference
5) DH - altitude difference
6) DV (1) - vertical velocity difference
7) DV (2) - east velocity difference
8) DV (3) - north velocity difference
9) DETA(1)- roll difference
10) DETA(2)- pitch difference
11) DETA(3)- heading difference

G. Sample SCOPE Control Cards (assuming TAPE 5 has been equated to INPUT)

```
ATTACH, PROG,NUMSIM,CY=8
VSN,TAPE20=L05370.
LABEL,TAPE20,R,L=TENMINJOB,D=HD,NORING
PROG.
DISPOSE, TAPE6,PR=IBD
*EOR
simulator input
```

H. RESTART HOOK

The Simulator has been designed with the potential for the easy inclusion of a checkpoint/restart facility. The key to this facility is the fact that all variables that are carried "from navigation cycle to navigation cycle" are stored in common blocks.

The checkpointing can therefore be accomplished by adding a subroutine. The subroutine could be called periodically from the exec (one could control the timing with RSTRT, which, currently, is unused) and would write out (on a tape presumably) the contents of the COMMON blocks.

The restart procedure would therefore consist of

a) Read in an operator input describing the simulated time of which restart is to occur

b) Find, in the checkpoint file, the data corresponding to the time from step a). Read this data in to the COMMON blocks

c) Find, in the binary input tape, the time corresponding to the time from step a). Leave the input tape positioned after this record

d) Resume computations at the beginning of the next navigation cycle.

R-977

VOLUME III

DISTRIBUTION LIST

R. Bumstead
W. Caffery
A. Ciccolo
G. Coate
R. Crisp
K. Daly
M. Dare
W. Delaney
W. Denhard
J. DiSorbo
K. Fertig
J. Fish
J. Harper
R. Harris
J. Hursh
P. Kampion
J. Kishel
W. Koenigsberg
R. Leger
R. Marshall
B. McCoy
W. McFarland
R. Nurse
P. Peck
T. Reed
D. Riegsecker
G. Schmidt (2)
J. Sciegienny (2)
R. Setterlund
L. W. Torrey
R. Wexler
TIC (5)

Air Force Avionics Laboratory (20)
Attn: RWA-3
Wright-Patterson Air Force Base,
Ohio 45433

Air Force Avionics Laboratory (20)
Attn: Capt. E. Harrington, RWA-2
Wright-Patterson Air Force Base,
Ohio 45433